# Informatyka 2 (ES1E3017)

Politechnika Białostocka - Wydział Elektryczny

Elektrotechnika, semestr III, studia stacjonarne I stopnia

Rok akademicki 2020/2021

Wykład nr 1 (06.10.2020)

dr inż. Jarosław Forenc



## Dane podstawowe


- dr inż. Jarosław Forenc
- Politechnika Białostocka, Wydział Elektryczny,
  Katedra Elektrotechniki, Energoelektroniki i Elektroenergetyki
  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 - dodatkowe materiały do pracowni

- Konsultacje
  - wtorek, 16:00-17:00, WE-204 / Teams
  - piątek, 09:00-11:00, 14:00-15:00, WE-204 / Teams
  - sobota, 16:00 -17:00, Teams (zaoczne)
  - niedziela, 14:00 - 15:30, Teams (zaoczne)



## Program wykładu (1/2)

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



## Program wykładu (2/2)

5. Operacje wejścia-wyjścia w języku C: znakowe, łańcuchowe, sformatowane, rekordowe. Pliki tekstowe i binarne.
6. System operacyjny. Zarządzanie procesami i dyskowymi operacjami wejścia-wyjścia. Systemy plików (FAT, NTFS, ext). Zarządzanie pamięcią operacyjną.
7. Sieci komputerowe. Topologie i media transmisyjne. Model referencyjny ISO/OSI i model protokołu TCP/IP.
8. Zaliczenie wykładu.

# 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.J. Deitel, H. Deitel: „Język C. Solidna wiedza w praktyce. Wydanie VIII”. Helion, Gliwice, 2020.
4. S.G. Kochan: „Język C. Kompendium wiedzy. Wydanie IV”. Helion, Gliwice, 2015.
5. R. Reese: „Wskaźniki w języku C. Przewodnik”. Helion, Gliwice, 2014.
6. R. Kawa, J. Lembas: „Wykłady z informatyki. Wstęp do informatyki”. PWN, Warszawa 2017.



# Literatura (2/2)

7. G. Coldwin: „Zrozumieć programowanie”. PWN, Warszawa, 2015.
8. A.S. Tanenbaum, H. Bos: „Systemy operacyjne. Wydanie IV”. Helion, Gliwice, 2015.
9. W. Stallings: „Systemy operacyjne. Architektura, funkcjonowanie i projektowanie. Wydanie IX”. Helion, Gliwice, 2018.
10. A.S. Tanenbaum, D.J. Wetherall: „Sieci komputerowe. Wydanie V”. Helion, Gliwice, 2012.
11. J. Kurose, K. Ross: „Sieci komputerowe. Ujęcie całościowe. Wydanie VII”. Helion, Gliwice, 2018.



# Efekty uczenia się i system ich oceniania

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

| | |
|---|---|
| **EU1** | zna w stopniu podstawowym zasady stosowania tablic, struktur, funkcji, plików i wskaźników w programach w języku C |
| **EU2** | opisuje podstawowe zadania systemu operacyjnego oraz strukturę sieci komputerowych |



# Zaliczenie wykładu - efekty uczenia się (EU1)

- 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 uczenia się (EU1)

- 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 uczenia się (EU1)

- 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 uczenia się (EU1)

- 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 uczenia się (EU2)

- 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 uczenia się (EU2)

- 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

- Sprawdzian pisemny na terenie Uczelni w terminie ustalonym ze studentami (po ósmym tygodniu semestru)
- Na zaliczeniu oceniane będą dwa efekt uczenia się (EU1, EU2)
- Za każdy efekt uczenia się 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 |

- Każdy efekt uczenia się musi być zaliczony na ocenę pozytywną (min. 51 punktów)



# Zaliczenie wykładu

- Prowadzący zajęcia może przyznawać dodatkowe punkty za aktywność na wykładzie
- Ocena końcowa wyznaczana jest na podstawie sumy otrzymanych punktów:

| 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 |

- W przypadku czasowego ograniczenia lub zawieszenia funkcjonowania Uczelni sprawdzian zaliczający wykład odbędzie się z wykorzystaniem technik zdalnego nauczania



# Terminy zajęć

- Wykład nr 1 - 06.10.2020
- Wykład nr 2 - 13.10.2020
- Wykład nr 3 - 20.10.2020
- Wykład nr 4 - 27.10.2020
- Wykład nr 5 - 03.11.2020
- Wykład nr 6 - 10.11.2020
- Wykład nr 7 - 17.11.2020
- Wykład nr 8 - 24.11.2020 (1h, 10:15-11:00)

# Plan wykładu nr 1

- Tablice w języku C
  - jednowymiarowe - wektory (przypomnienie)
  - dwuwymiarowe - macierze
  - wielowymiarowe
- Tablice o zmiennym rozmiarze (VLA)
- Łańcuchy znaków w języku C
  - implementacja, deklaracja, inicjalizacja
  - stała znakowa
  - wyświetlenie i wczytanie tekstu
  - plik nagłówkowy string.h



# 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 |

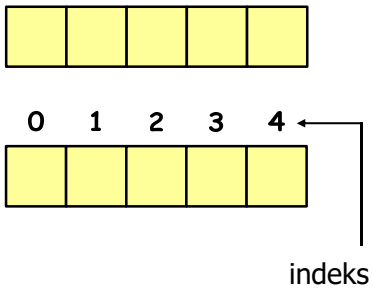
| 1.2 | 2.5 | 2.0 | 10.0 |
|---|---|---|---|
| -0.1 | 4.3 | 6.2 | -5.1 |
| 0.0 | 12.2 | 4.1 | -2.2 |



# Język C - deklaracja tablica jednowymiarowej

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

rozmiar
nazwa
typ elementów




- 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];
```

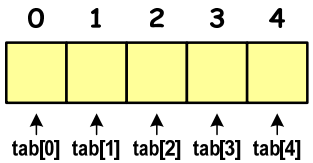


# Język C - odwołania do elementów tablicy

[ ] - dwuargumentowy operator indeksowania

```
tab[1];
```

indeks elementu (drugi argument operatora)
nazwa tablicy (pierwszy argument operatora)




- 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

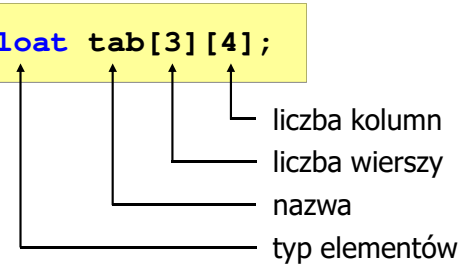
```
int tab[] = {1,2,3,4,5};
```

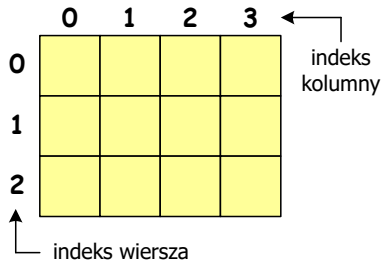
| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |



# Język C - deklaracja tablica dwuwymiarowej

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



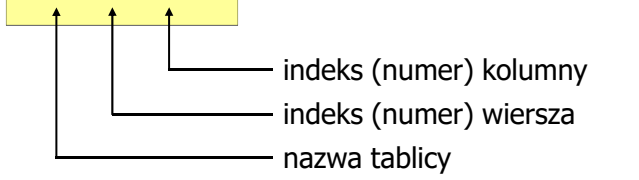

- Rozmiar tablicy (liczba wierszy, liczba kolumn) to wartość:
  - całkowita, dodatnia
  - znana na etapie kompilacji programu
    (stała liczbowa: 5, #define N 5, const int n = 5;)

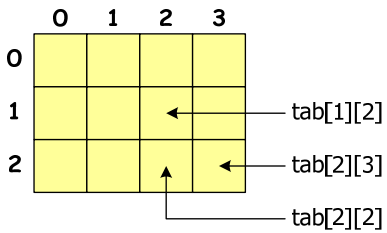


# Język C - odwołania do elementów macierzy

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

[ ] - dwuargumentowy operator indeksowania




- Indeks:
  - 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};
```

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 |

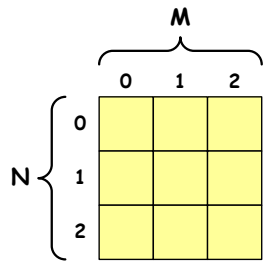


# Język C - operacje na macierzy

```
#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;
```

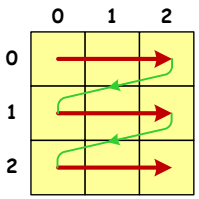




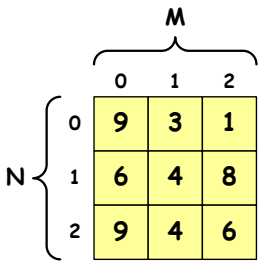

# Język C - operacje na macierzy

```
/* 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



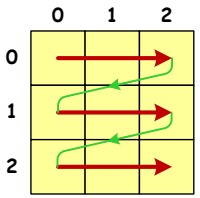



# Język C - operacje na macierzy

```
/* 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
```



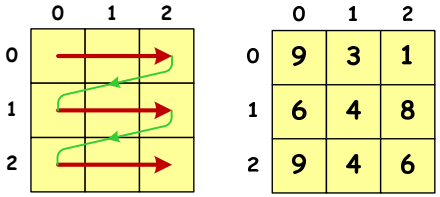

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 9 | 3 | 1 |
| 1 | 6 | 4 | 8 |
| 2 | 9 | 4 | 6 |

# 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
```



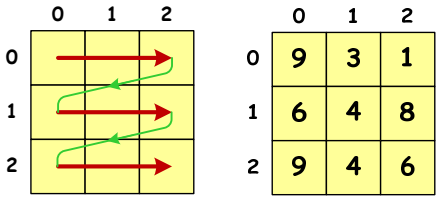



# Język C - operacje na macierzy

```
/* 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
```



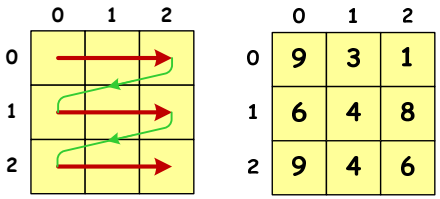



# Język C - operacje na macierzy

```
/* 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);
}
```

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



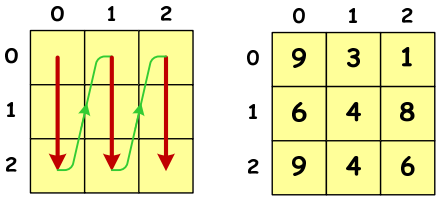



# Język C - operacje na macierzy

```
/* 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);
}
```

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



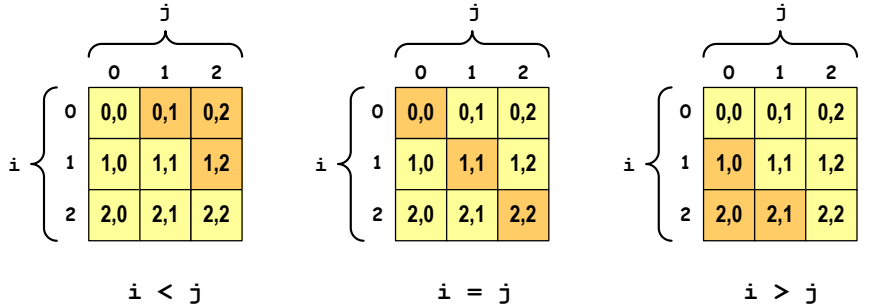
# Język C - operacje na macierzy

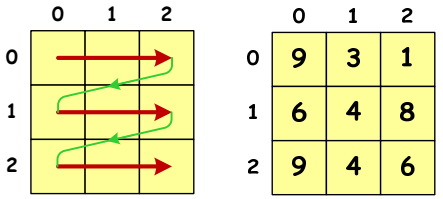
```
/* sumy elementow nad, na i ponizej przekatnej */
suma = suma1 = suma2 = 0;
for (i=0; i<N; i++)
    for (j=0; j<M; j++)
    {
        if (i < j) suma1+=tab[i][j];  /* nad */
        if (i > j) suma2+=tab[i][j];  /* pod */
        if (i == j) suma+=tab[i][j];  /* na  */
    }
printf("Suma nad: %d\n",suma1);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




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



# Język C - tablice wielowymiarowe

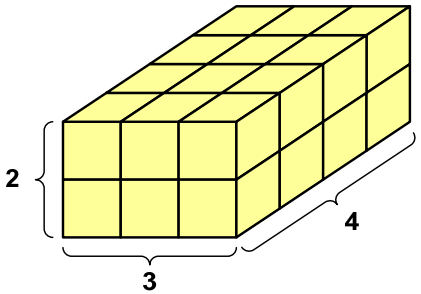
- Deklaracja tablicy wielowymiarowej

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

- Deklaracja tablicy trójwymiarowej

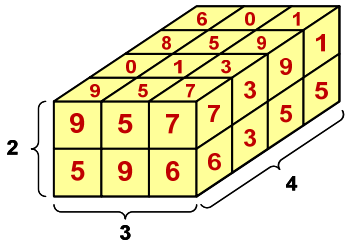
```
int tab[4][2][3];
```

- 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

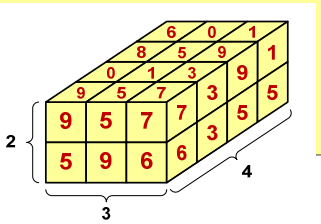
```
    for(z=0; z<Z; z++)
    {
        for(y=0; y<Y; y++)
        {
            for(x=0; x<X; x++)
                printf("%3d",tab[z][y][x]);
            printf("\n");
        }
        printf("\n");
    }
    return 0;
}
```

```
  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 - 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



# Język C - 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;
}
```



# Język C - tablice VLA (VC++ 2008)

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

int main(void)
{
    int n, i;

    printf("Rozmiar we                );
    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

# Język C - tablice VLA (Dev-C++, Code::Blocks)

```c
#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

```c
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 - 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



# Język C - modularność programu

- Przykład

```c
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];
}
```

```c
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];
```

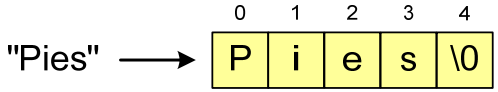
- Zamiast jednej pętli for stosowane są trzy pętle

# 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)

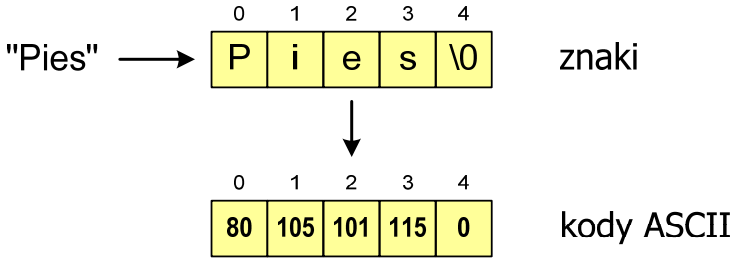



- Ostatni znak (\0, liczba zero, znak zerowy) oznacza koniec napisu



# Język C - łańcuchy znaków

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






# 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

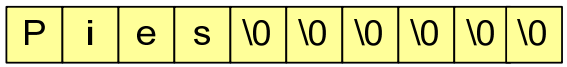


# Język C - inicjalizacja łańcucha znaków

- 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




```
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

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



# Język C - wyświetlenie tekstu

- Wyświetlenie tekstu funkcją printf() wymaga specyfikatora %s

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

```
Osoba: [Jan Kowalski]
```

- W specyfikatorze %s: szerokość określa szerokość pola, zaś precyzja - liczbę pierwszych znaków z łańcucha

```
char napis[15] = "Jan Kowalski";
printf("[%10.6s]\n", napis);
```

```
[    Jan Ko]
```

# Język C - wyświetlenie tekstu

- Do wyświetlenia tekstu można zastosować funkcję puts()

| puts() | int puts(const char *s); |
|---|---|

- Funkcja puts() wypisuje na stdout (ekran) zawartość łańcucha znakowego (ciąg znaków zakończony znakiem '\0'), zastępując znak '\0' znakiem '\n'

```
char napis[15] = "Jan Kowalski";
puts(napis);
```

```
Jan Kowalski
```



# Język C - wyświetlenie tekstu

- Wyświetlenie znaku funkcją printf() wymaga specyfikatora %c

```
char zn = 'x';
printf("Znak to: [%c]\n", zn);
```

```
Znak to: [x]
```



# Język C - wyświetlenie tekstu

- Łańcuch znaków jest zwykłą tablicą - można więc odwoływać się do jej pojedynczych elementów

```
char txt[15] = "Ola ma laptopa";

printf("Znaki: ");
for (int i=0; i<15; i++) printf("%c ",txt[i]);
```

```
Znaki: O l a   m a   l a p t o p a
```

```
printf("Kody:  ");
for (int i=0; i<15; i++) printf("%d ",txt[i]);
```

```
Kody:  79 108 97 32 109 97 32 108 97 112 116 111 112 97 0
```



# Język C - wczytanie tekstu

- Do wczytania tekstu funkcją scanf() stosowany jest specyfikator %s

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

brak znaku &

- W specyfikatorze formatu %s można podać szerokość

```
char napis[15];
scanf("%10s", napis);
```

- W powyższym przykładzie scanf() zakończy wczytywanie tekstu po pierwszym białym znaku (spacja, tabulacja, enter) lub w momencie pobrania 10 znaków

# Język C - wczytanie tekstu

- 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()

| gets() | char *gets(char *s); |
|---|---|

- Funkcja gets() wprowadza wiersz (ciąg znaków zakończony '\n') ze strumienia stdin (klawiatura) i umieszcza w obszarze pamięci wskazywanym przez wskaźnik s zastępując '\n' znakiem '\0'

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



# 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)

```
#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 - 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)

```
char txt[3][15] = {"Programowanie",
                   "nie jest","trudne"};
for (int i=0; i<3; i++)
{
    for (int j=0; j<6; j++)
       printf("%c",txt[i][j]);
    printf("\n");
}
```

```
Progra
nie je
trudne
```

| | 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

- 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!