

# Informatyka 1 (ES1E2009)

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

## Wykład nr 4 (07.04.2020)

dr inż. Jarosław Forenc

## Plan wykładu nr 4

- Język C - pętle
  - pętla for
  - operatory ++ i --
  - pętle while i do...while
- Język C - tablice jednowymiarowe (wektory)
  - deklaracja, odwołania do elementów, inicjalizacja tablicy
  - generator liczb pseudolosowych
  - operacje na wektorze

## Język C - suma kolejnych 10 liczb: 1+2+...+10

```
#include <stdio.h>

int main(void)
{
    int suma;

    suma = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10;

    printf("Suma wynosi: %d\n", suma);

    return 0;
}
```

Suma wynosi: 55

## Język C - suma kolejnych 100 liczb: 1+2+...+100

```
#include <stdio.h>

int main(void)
{
    int suma=0, i;

    for (i=1; i<=100; i=i+1)
        suma = suma + i;

    printf("Suma wynosi: %d\n", suma);

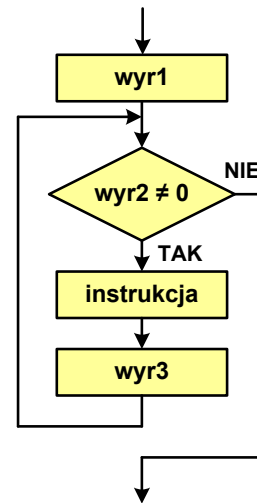
    return 0;
}
```

Suma wynosi: 5050

## Język C - pętla for

```
for (wyr1; wyr2; wyr3)  
instrukcja
```

- **wyr1, wyr2, wyr3** - dowolne wyrażenia w języku C
- Instrukcja:
  - **prosta** - jedna instrukcja zakończona średnikiem
  - **złożona** - jedna lub kilka instrukcji objętych nawiasami klamrowymi



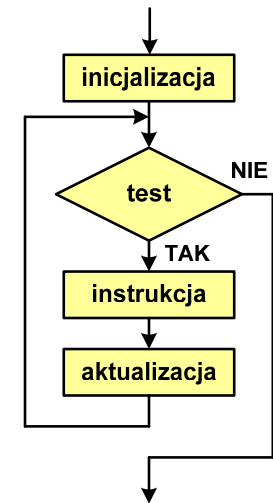
## Język C - pętla for

- Najczęściej stosowana postać pętli **for**

```
int i;  
for (i = 0; i < 10; i = i + 1)  
instrukcja
```

- Instrukcja zostanie wykonana 10 razy (dla  $i = 0, 1, 2, \dots, 9$ )
- Funkcje pełnione przez wyrażenia

```
for (inicjalizacja; test; aktualizacja)  
instrukcja
```



## Język C - pętla for (wyświetlenie tekstu)

```
#include <stdio.h>  
  
int main(void)  
{  
    int i;  
  
    for (i=0; i<5; i=i+1)  
        printf("Programowanie nie jest trudne\n");  
  
    return 0;  
}
```

```
Programowanie nie jest trudne  
Programowanie nie jest trudne  
Programowanie nie jest trudne  
Programowanie nie jest trudne  
Programowanie nie jest trudne
```

## Język C - pętla for (suma liczb: 1 + 2 + ... + N)

```
#include <stdio.h>  
#define N 1234  
  
int main(void)  
{  
    int i, suma=0;  
  
    for (i=1; i<=N; i++)  
        suma = suma + i;  
  
    printf("Suma %d liczb to %d\n", N, suma);  
  
    return 0;  
}
```

```
Suma 1234 liczb to 761995
```

## Język C - pętla for (przykłady)

```
for (i=0; i<10; i++)  
    printf("%d ",i);
```

0 1 2 3 4 5 6 7 8 9

```
for (i=0; i<10; i++)  
    printf("%d ",i+1);
```

1 2 3 4 5 6 7 8 9 10

```
for (i=1; i<=10; i++)  
    printf("%d ",i);
```

1 2 3 4 5 6 7 8 9 10

## Język C - pętla for (przykłady)

```
for (i=1; i<10; i=i+2)  
    printf("%d ",i);
```

1 3 5 7 9

```
for (i=10; i>0; i--)  
    printf("%d ",i);
```

10 9 8 7 6 5 4 3 2 1

```
for (i=-9; i<=9; i=i+3)  
    printf("%d ",i);
```

-9 -6 -3 0 3 6 9

## Język C - pętla for (break, continue)

- W pętli `for` można stosować instrukcje skoku: `break` i `continue`

```
int i;  
for (i=1; i<10; i++)  
{  
    if (i%2==0)  
        continue;  
    if (i%7==0)  
        break;  
    printf("%d\n",i);  
}
```

□ `continue` przerywa bieżącą iterację i przechodzi do obliczania `wyr3`

□ `break` przerywa wykonywanie pętli

1 3 5

## Język C - pętla for (najczęstsze błędy)

- Postawienie średnika na końcu pętli `for`

```
int i;  
for (i=0; i<10; i++);  
printf("%d ",i);
```

10

- Przecinki zamiast średników pomiędzy wyrażeniami

```
int i;  
for (i=0, i<10, i++)  
    printf("%d ",i);
```

*Błąd kompilacji!*

error C2143: syntax error : missing ';' before ')'

## Język C - pętla for (najczęstsze błędy)

- Błędny warunek - brak wykonania instrukcji

```
int i;  
for (i=0; i>10; i++)  
    printf("%d ", i);
```

- Błędny warunek - pętla nieskończona

```
int i;  
for (i=1; i>0; i++)  
    printf("%d ", i);
```

```
1 2 3 4 5 6 7 8 9 ...
```

## Język C - pętla nieskończona

```
for (wyr1; wyr2; wyr3)  
    instrukcja
```

- Wszystkie wyrażenia (**wyr1**, **wyr2**, **wyr3**) w pętli for są opcjonalne

```
for ( ; ; )  
    instrukcja
```

- pętla nieskończona

- W przypadku braku **wyr2** przyjmuje się, że jest ono **prawdziwe**

## Język C - zagnieżdżanie pętli for

- Jako instrukcja w pętli **for** może występować kolejna pętla **for**

```
int i, j;  
for (i=1; i<=3; i++)           // pętla zewnętrzna  
    for (j=1; j<=2; j++)       // pętla wewnętrzna  
        printf("i: %d  j: %d\n", i, j);
```

```
i: 1  j: 1  
i: 1  j: 2  
i: 2  j: 1  
i: 2  j: 2  
i: 3  j: 1  
i: 3  j: 2
```

## Język C - operator inkrementacji (++)

- Jednoargumentowy operator **++** zwiększa wartość zmiennej o 1 (nie wolno stosować go do wyrażeń)
- Operator **++** może występować jako przedrostek lub przyrostek

Zapis	Nazwa	Znaczenie
<b>++x</b>	preinkrementacji	wartość zmiennej jest modyfikowana przed jej użyciem
<b>x++</b>	postinkrementacji	wartość zmiennej jest modyfikowana po użyciu jej poprzedniej wartości

## Język C - operator inkrementacji (++)

### Przykład

```
int x = 1, y;  
y = 2 * ++x;
```

```
int x = 1, y;  
y = 2 * x++;
```

### Kolejność operacji

```
++x      x = 2  
2 * ++x  2 * 2  
y = 2 * ++x  y = 4
```

```
2 * x      2 * 1  
y = 2 * x  y = 2  
x++       x = 2
```

### Wartości zmiennych

```
x = 2    y = 4
```

```
x = 2    y = 2
```

## Język C - operator inkrementacji (++)

### Miejsce umieszczenia operatora ++ nie ma znaczenia w przypadku instrukcji typu:

```
x++;  
++x;
```

równoważne

```
x = x + 1;
```

### Nie należy stosować operatora ++ do zmiennych pojawiających się w wyrażeniu więcej niż jeden raz

```
x = x++;  
x = ++x;
```

### Zgodnie ze standardem języka C wynik powyższych instrukcji jest **niezdefiniowany**

## Język C - operator dekrementacji (--)

- Jednoargumentowy operator -- zmniejsza wartość zmiennej o 1 (nie wolno stosować go do wyrażeń)
- Operator -- może występować jako przedrostek lub przyrostek

Zapis	Nazwa	Znaczenie
--x	predekrementacji	wartość zmiennej jest modyfikowana przed jej użyciem
x--	postdekrementacji	wartość zmiennej jest modyfikowana po użyciu jej poprzedniej wartości

## Język C - priorytet operatorów ++ i --

Priorytet	Operator / opis
1	++ -- (przyrostki) () [] . ->
2	++ -- (przedrostki) sizeof (typ) + - ! ~ * & (jednoargumentowe)
3	* / %
4	+ - (dwuargumentowe)
5	<< >>
6	< > <= >=
7	== !=
8	& (bitowy)
9	^

## Język C - miesięczny kalendarz

- Napisz program wyświetlający miesięczny kalendarz. Wczytaj liczbę dni w miesiącu i dzień tygodnia, od którego zaczyna się miesiąc.
- Przykład działania programu:

Liczba dni w miesiącu: 31  
Pierwszy dzień tygodnia (1-Pn, 2-Wt, ...): 4

```
-----  
Pn Wt Sr Cz Pt So  N  
          1  2  3  4  
 5  6  7  8  9 10 11  
12 13 14 15 16 17 18  
19 20 21 22 23 24 25  
26 27 28 29 30 31
```

## Język C - miesięczny kalendarz

```
#include <stdio.h>  
  
int main()  
{  
    int ile_dni, dzien_tyg, i;  
  
    printf("Liczba dni w miesiacu: "); scanf("%d",&ile_dni);  
    printf("Pierwszy dzien tygodnia (1-Pn, 2-Wt, ...): ");  
    scanf("%d",&dzien_tyg);  
  
    printf("-----\n");  
    printf(" Pn Wt Sr Cz Pt So  N\n");  
  
    for (i=1; i<dzien_tyg; i++) printf("  ");  
    for (i=0; i<ile_dni; i++)  
    {  
        printf("%3d",i+1);  
        if ((i+dzien_tyg)%7==0) printf("\n");  
    }  
    printf("\n"); return 0;  
}
```

## Język C - miesięczny kalendarz

```
#include <stdio.h>  
  
int main()  
{  
    int ile_dni, dzien_tyg, i;  
  
    printf("Liczba dni w miesiacu: "); scanf("%d",&ile_dni);  
    printf("Pierwszy dzien tygodnia (1-Pn, 2-Wt, ...): ");  
    scanf("%d",&dzien_tyg);  
  
    printf("-----\n");  
    printf(" Pn Wt Sr Cz Pt So  N\n");  
  
    for (i=1; i<dzien_tyg; i++) printf("  ");  
    for (i=0; i<ile_dni; i++)  
    {  
        printf("%3d",i+1);  
        if ((i+dzien_tyg)%7==0) printf("\n");  
    }  
    printf("\n"); return 0;  
}
```

```
Liczba dni w miesiacu: 30  
Pierwszy dzien tygodnia (1-Pn, 2-Wt, ...): 5  
-----  
Pn Wt Sr Cz Pt So  N  
          1  2  3  
 4  5  6  7  8  9 10  
11 12 13 14 15 16 17  
18 19 20 21 22 23 24  
25 26 27 28 29 30
```

## Język C - pierwiastek kwadratowy

```
#include <stdio.h>  
#include <math.h>  
  
int main(void)  
{  
    float x, y;  
  
    printf("Podaj liczbe: ");  
    scanf("%f",&x);  
  
    if (x>=0)  
    {  
        y = sqrt(x);  
        printf("Pierwiastek liczby: %f\n",y);  
    }  
    else  
        printf("Blad! Liczba ujemna\n");  
  
    return 0;  
}
```

```
Podaj liczbe: -3  
Blad! Liczba ujemna
```

```
Podaj liczbe: 3  
Pierwiastek liczby: 1.732051
```

## Język C - pierwiastek kwadratowy (pętla while)

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

int main(void)
{
    float x, y;

    printf("Podaj liczbe: ");
    scanf("%f", &x);
    while (x<0)
    {
        printf("Blad! Liczba ujemna\n\n");
        printf("Podaj liczbe: ");
        scanf("%f", &x);
    }
    y = sqrt(x);
    printf("Pierwiastek liczby: %f\n",y);

    return 0;
}
```

```
Podaj liczbe: -3
Blad! Liczba ujemna

Podaj liczbe: -5
Blad! Liczba ujemna

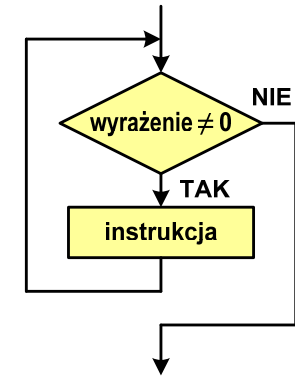
Podaj liczbe: 3
Pierwiastek liczby: 1.732051
```

## Język C - pętla while

```
while (wyrażenie)
    instrukcja
```

- „dopóki wyrażenie w nawiasach jest prawdziwe wykonuj instrukcję”

- Wyrażenie w nawiasach:
  - prawdziwe - gdy jego wartość jest różna od zera
  - falszywe - gdy jego wartość jest równa zero
- Jako wyrażenie najczęściej stosowane jest **wyrażenie logiczne**



## Język C - pętla while

```
while (wyrażenie)
    instrukcja
```

- Instrukcja:
  - prosta - jedna instrukcja zakończona średnikiem
  - złożona - jedna lub kilka instrukcji objętych nawiasami klamrowymi

```
int x = 10;
while (x>0)
    x = x - 1;
```

```
int x = 10;
while (x>0)
{
    printf("%d\n", x);
    x = x - 1;
}
```

## Język C - suma liczb dodatnich

```
#include <stdio.h>

int main(void)
{
    int x, suma = 0;

    printf("Podaj liczbe: ");
    scanf("%d", &x);

    while(x>0)
    {
        suma = suma + x;
        printf("Podaj liczbe: ");
        scanf("%d", &x);
    }
    printf("Suma liczb: %d\n", suma);

    return 0;
}
```

```
Podaj liczbe: 4
Podaj liczbe: 8
Podaj liczbe: 2
Podaj liczbe: 3
Podaj liczbe: 5
Podaj liczbe: -2
Suma liczb: 22
```

## Język C - pętla while

- Program pokazany na poprzednim slajdzie zawiera typowy schemat przetwarzania danych z wykorzystaniem pętli **while**

```
printf("Podaj liczbę: ");  
scanf("%d", &x);  
  
while (x>0)  
{  
    suma = suma + x;  
    printf("Podaj liczbę: ");  
    scanf("%d", &x);  
}
```

wczytanie danych

operacje na danych

wczytanie danych

- Dane mogą być wczytywane z klawiatury, pliku, itp.

## Język C - pętla while (break, continue)

- break** i **continue** są to instrukcje skoku

```
int x=0;  
while (x<10)  
{  
    x++;  
    if (x%2==0)  
        continue;  
    if (x%5==0)  
        break;  
    printf("%d\n", x);  
}
```

□ **continue** przerywa bieżącą iterację

□ **break** przerywa wykonywanie pętli

## Język C - pętla while (najczęstsze błędy)

- Postawienie średnika po wyrażeniu w nawiasach powoduje powstanie pętli nieskończonej - program zatrzymuje się na pętli

```
int x = 10;  
while (x>0);  
    printf("%d ", x--);
```



- Brak aktualizacji zmiennej powoduje także powstanie pętli nieskończonej - program wyświetla wielokrotnie tę samą wartość

```
int x = 10;  
while (x>0)  
    printf("%d ", x);
```

10 10 10 10 10 ...

## Język C - pętla while (pętla nieskończona)

- W pewnych sytuacjach celowo stosuje się pętlę nieskończoną (np. w mikrokontrolerach)

```
while (1)  
{  
    instrukcja  
    instrukcja  
    ...  
}
```

- W układach mikroprocesorowych program działa aż do wyłączenia zasilania

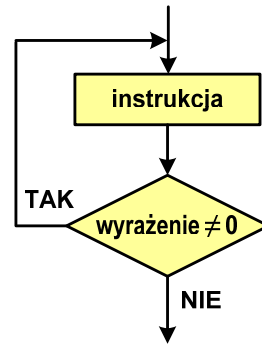


## Język C - pętla do ... while

```
do
    instrukcja
while (wyrażenie);
```

- „wykonuj instrukcję dopóki wyrażenie w nawiasach jest prawdziwe”

- Wyrażenie w nawiasach:
  - prawdziwe** - gdy jego wartość jest różna od zera
  - falszywe** - gdy jego wartość jest równa zero



## Język C - pętla do ... while

```
do
    instrukcja
while (wyrażenie);
```

- Instrukcja:
  - prosta** - jedna instrukcja zakończona średnikiem
  - złożona** - jedna lub kilka instrukcji objętych nawiasami klamrowymi

```
int x = 10;
do
    x = x - 1;
while (x>0);
```

```
int x = 10;
do
{
    printf("%d\n", x);
    x = x - 1;
}
while (x>0);
```

## Język C - pętla do ... while (break, continue)

- break** i **continue** są to instrukcje skoku

```
int x=0;

do
{
    x++;
    if (x%5==0)
        break;
    if (x%2==0)
        continue;
    printf("%d\n", x);
}
while (i<10);
```

- break** przerywa wykonywanie pętli
- continue** przerywa bieżącą iterację

## Język C - suma liczb < 100

```
#include <stdio.h>

int main(void)
{
    int x, suma = 0;

    do
    {
        printf("Podaj liczbe: ");
        scanf("%d", &x);
        suma = suma + x;
    }
    while (suma<100);

    printf("Suma liczb: %d\n", suma);

    return 0;
}
```

```
Podaj liczbe: 34
Podaj liczbe: 9
Podaj liczbe: 26
Podaj liczbe: -8
Podaj liczbe: 67
Suma liczb: 128
```

## Język C - operacje na dużej ilości danych

```
#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 - operacje na dużej ilości danych

```
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 - 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 - tablica elementów

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

wektor

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

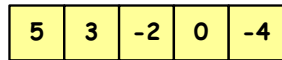
macierz

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

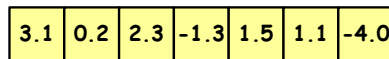
1.2	2.5	2.0	10.0
-0.1	4.3	6.2	-5.1
0.0	12.2	4.1	-2.2

## Język C - tablica jednowymiarowa

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



- liczby całkowite

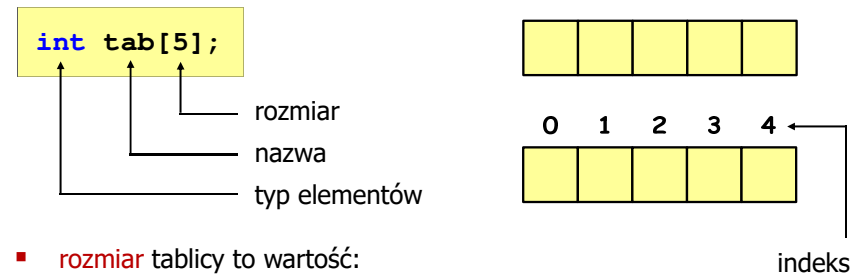


- liczby rzeczywiste



- znaki

## 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`;)

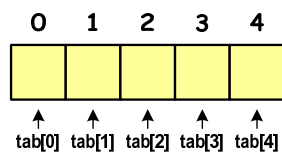
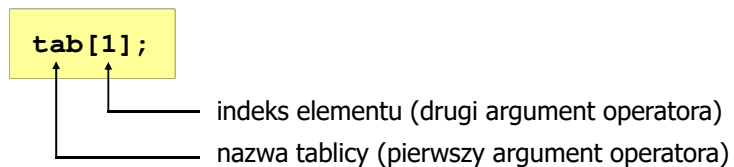
`int tab[5];`

`int tab[N];`

`int tab[n];`

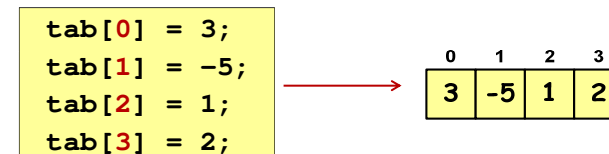
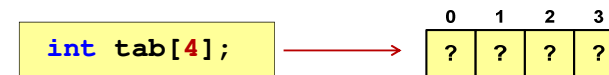
## Język C - odwołania do elementów tablicy

`[ ]` - 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



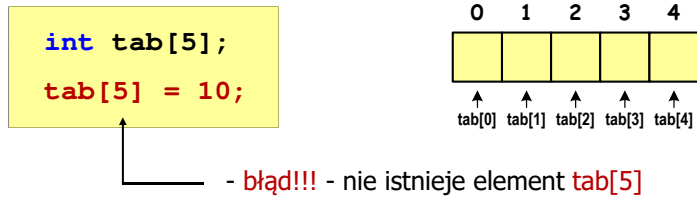
- Każdy element tablicy traktowany jest tak samo jak zmienna typu `int`

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

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

## Język C - odwołania do elementów tablicy

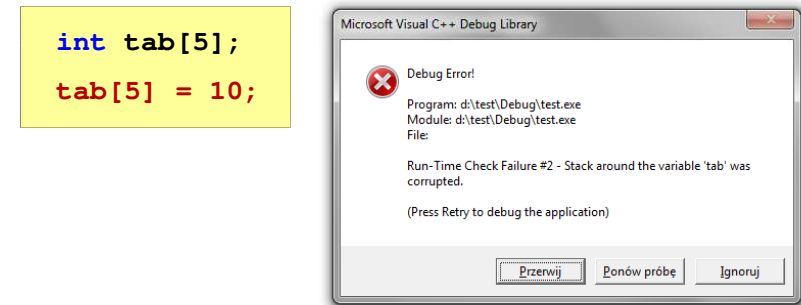
- Przy odwołaniach do elementów tablicy kompilator nie sprawdza poprawności indeksów



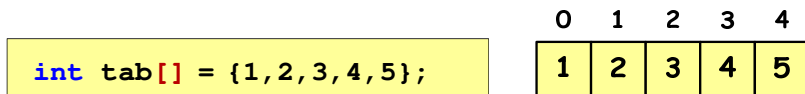
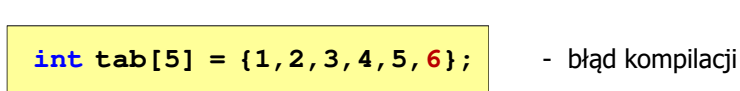
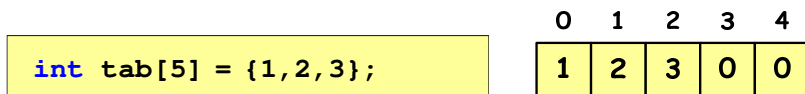
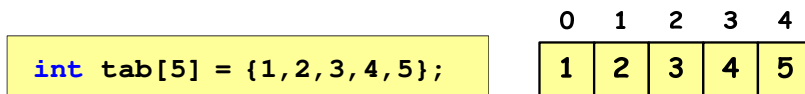
- Kompilator nie zasygnalizuje błędu
- Program wykona operację
- Środowisko programistyczne może zasygnalizować problem

## Język C - odwołania do elementów tablicy

- Przy odwołaniach do elementów tablicy kompilator nie sprawdza poprawności indeksów



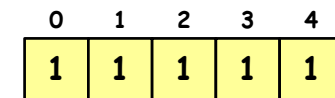
## Język C - inicjalizacja tablicy jednowymiarowej



## 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 - 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;

    /* 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
7	12	1	16	1	11	14	5	19	8

## 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:
7 12 1 16 1 11 14 5 19 8
```

0	1	2	3	4	5	6	7	8	9
7	12	1	16	1	11	14	5	19	8

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:
8 19 5 14 11 1 16 1 12 7
```

0	1	2	3	4	5	6	7	8	9
7	12	1	16	1	11	14	5	19	8

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: 1

0	1	2	3	4	5	6	7	8	9
7	12	1	16	1	11	14	5	19	8

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: 2 4

0	1	2	3	4	5	6	7	8	9
7	12	1	16	1	11	14	5	19	8

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: 94, srednia: 9.4

0	1	2	3	4	5	6	7	8	9
7	12	1	16	1	11	14	5	19	8

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: 4

0	1	2	3	4	5	6	7	8	9
7	12	1	16	1	11	14	5	19	8

N = 10

**Koniec wykładu nr 4**

**Dziękuję za uwagę!**