Informatyka 1 (ES1E2009)

Politechnika Białostocka - Wydział Elektryczny

Elektrotechnika, semestr II, studia stacjonarne I stopnia

Rok akademicki 2021/2022

Wykład nr 5 (17.05.2022)

dr inż. Jarosław Forenc

Plan wykładu nr 5

- Język C - tablice jednowymiarowe (wektory)
  - deklaracja, odwołania do elementów, inicjalizacja tablicy
  - generator liczb pseudolosowych, operacje na wektorze
- Kodowanie liczb
  - NKB, BCD, kod 2 z 5, kod Graya
- Reprezentacja liczb całkowitych
  - liczby bez znaku
  - liczby ze znakiem (ZM, U1, U2)
- Reprezentacja zmiennoprzecinkowa
  - zapis, postać znormalizowana
  - zakres liczb zmiennoprzecinkowych

Język C - tablica elementów

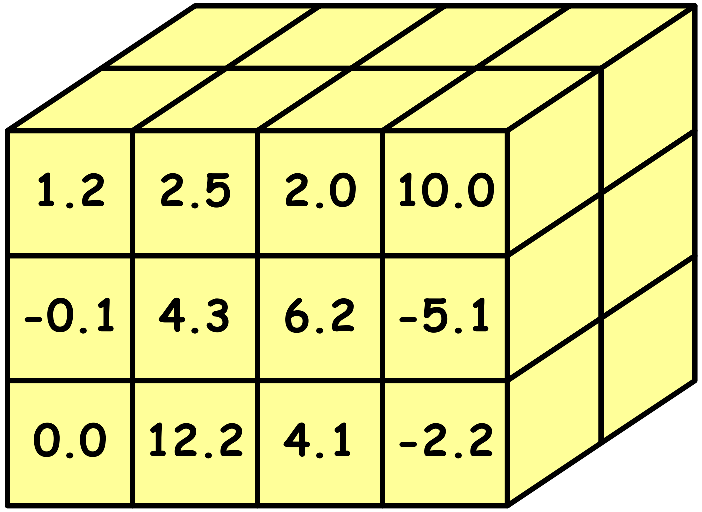
- Tablica - ciągły obszar pamięci, w którym umieszczone są elementy tego samego typu

wektor

5	3	-2	1	-4

macierz

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



Język C - 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

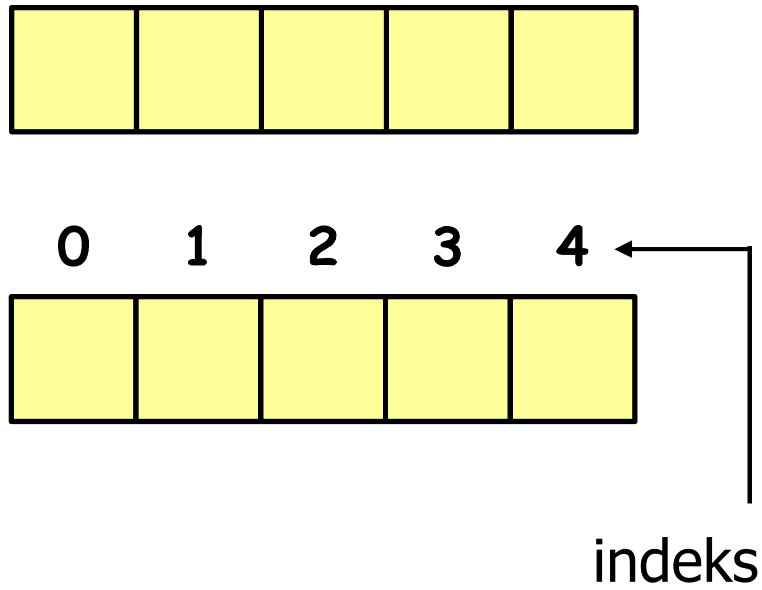
a	Z	x	&	M	+

- znaki

Język C - deklaracja tablicy jednowymiarowej

```c
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;)

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

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

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

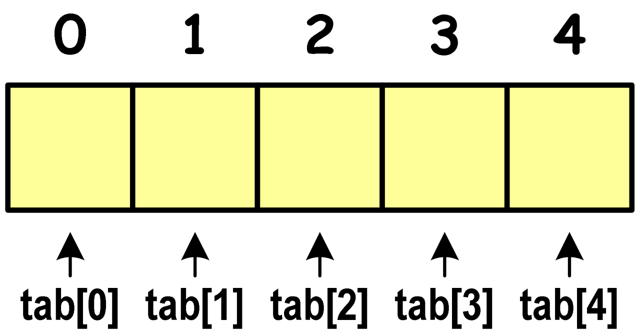
Język C - odwołania do elementów tablicy

[ ] - 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 - odwołania do elementów tablicy

```c
int tab[4];
```

0	1	2	3
?	?	?	?

```c
tab[0] = 3;
tab[1] = -5;
tab[2] = 1;
tab[3] = 2;
```

0	1	2	3
3	-5	1	2

- Każdy element tablicy traktowany jest jak zmienna typu int

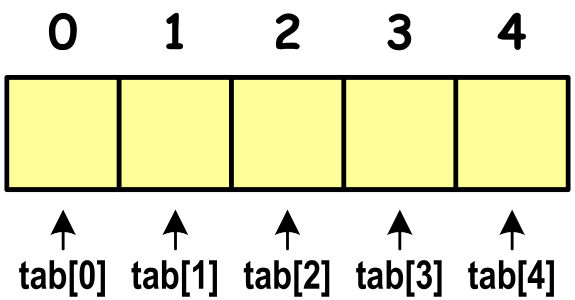
```c
printf("%d",tab[0]);
```

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

Język C - odwołania do elementów tablicy

- Przy odwołaniach do elementów tablicy kompilator nie sprawdza poprawności indeksów

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



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

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

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

```c
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

```c
int tab[5], i;

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

Przykład: operacje na dużej ilości danych

```
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;
}
```

	0	1	2	3	4
U	5.0	10.0	15.0	20.0	25.0

	0	1	2	3	4
I	0.16	0.21	0.27	0.33	0.36

	0	1	2	3	4
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

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

```c
#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;
```

Język C - operacje na wektorze

```c
/* 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 kolejnosci:\n");
for (i=N-1; i>=0; i--)
    printf("%d  ",tab[i]);
printf("\n");
```

```
Elementy w odwrotnej kolejnosci:
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);
```

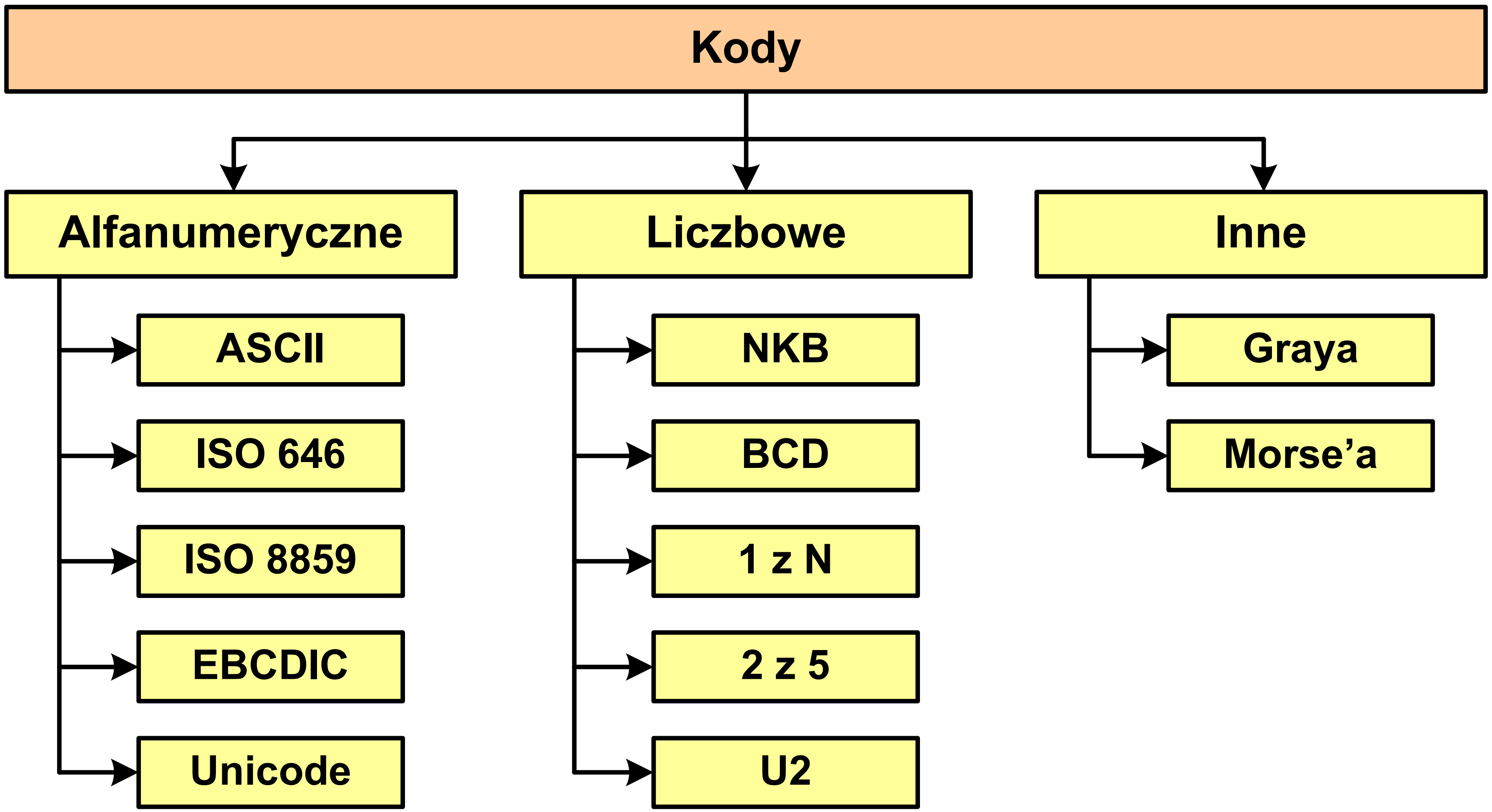
```
Liczba parzystych elementow: 6
```

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

N = 10

Kodowanie

- Kodowanie - proces przekształcania jednego rodzaju postaci informacji na inną postać



Kody liczbowe - Naturalny Kod Binarny (NKB)

- Jeżeli dowolnej liczbie dziesiętnej przypiszemy odpowiadającą jej liczbę binarną, to otrzymamy naturalny kod binarny (NKB)

128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
7	6	5	4	3	2	1	0

Liczba dziesiętna	Kod NKB
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Liczba dziesiętna	Kod NKB
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Kody liczbowe - Kod BCD

- Binary-Coded Decimal - dziesiętny zakodowany dwójkowo
- BCD - sposób zapisu liczb polegający na zakodowaniu kolejnych cyfr liczby dziesiętnej w 4-bitowym systemie dwójkowym (NKB)

Cyfra dziesiętna	BCD
0	0000
1	0001
2	0010
3	0011
4	0100

Cyfra dziesiętna	BCD
5	0101
6	0110
7	0111
8	1000
9	1001

- W ogólnym przypadku kodowane są tylko znaki 0 ÷ 9
- Pozostałe kombinacje bitowe mogą być stosowane do kodowania znaku liczby lub innych znaczników.

Kody liczbowe - Kod BCD

- Przykład:

$$168_{(10)} = ?_{(BCD)}$$

$$\overbrace{0001}^{1}\ \overbrace{0110}^{6}\ \overbrace{1000}^{8}$$

$$168_{(10)} = 000101101000_{(BCD)}$$

$$1001\,|\,0101\,|\,0011_{(BCD)} = ?_{(10)}$$

$$\underbrace{1001}_{9}\ \underbrace{0101}_{5}\ \underbrace{0011}_{3}$$

$$100101010011_{(BCD)} = 953_{(10)}$$

- Zastosowania:
  - urządzenia elektroniczne z wyświetlaczem cyfrowym (np. kalkulatory, mierniki cyfrowe, kasy sklepowe, wagi)
  - przechowywania daty i czasu w BIOSie komputerów (także wczesne modele PlayStation 3)
  - zapis części ułamkowych kwot (systemy bankowe).

Kody liczbowe - Kod BCD: przechowywanie liczb

- Użycie 4 najmłodszych bitów jednego bajta, 4 starsze bity są ustawiane na jakąś konkretną wartość:
  - 0000
  - 1111 (np. kod EBCDIC, liczby $F0_{(16)} \div F9_{(16)}$)
  - 0011 (tak jak w ASCII, liczby $30_{(16)} \div 39_{(16)}$)
- Zapis dwóch cyfr w każdym bajcie (starsza na starszej połówce, młodsza na młodszej połówce) - jest to tzw. spakowane BCD
  - w przypadku liczby zapisanej na kilku bajtach, najmniej znacząca tetrada (4 bity) używane są jako flaga znaku
  - standardowo przyjmuje się 1100 ($C_{(16)}$) dla znaku plus (+) i 1101 ($D_{(16)}$) dla znaku minus (-), np.

$$127_{(10)} = 0001\ \ 0010\ \ 0111\ \ 1100 \quad (127C_{(16)})$$

$$-127_{(10)} = 0001\ \ 0010\ \ 0111\ \ 1101 \quad (127D_{(16)})$$

Kody liczbowe - Kod BCD

- Warianty kodu BCD:

Cyfra dziesiętna	BCD 8421	Excess-3	BCD 2421	BCD 84-2-1	IBM 1401 BCD 8421
0	0000	0011	0000	0000	1010
1	0001	0100	0001	0111	0001
2	0010	0101	0010	0110	0010
3	0011	0110	0011	0101	0011
4	0100	0111	0100	0100	0100
5	0101	1000	1011	1011	0101
6	0110	1001	1100	1010	0110
7	0111	1010	1101	1001	0111
8	1000	1011	1110	1000	1000
9	1001	1100	1111	1111	1001

- Podstawowy wariant: BCD 8421 (SBCD - Simple Binary Coded Decimal)

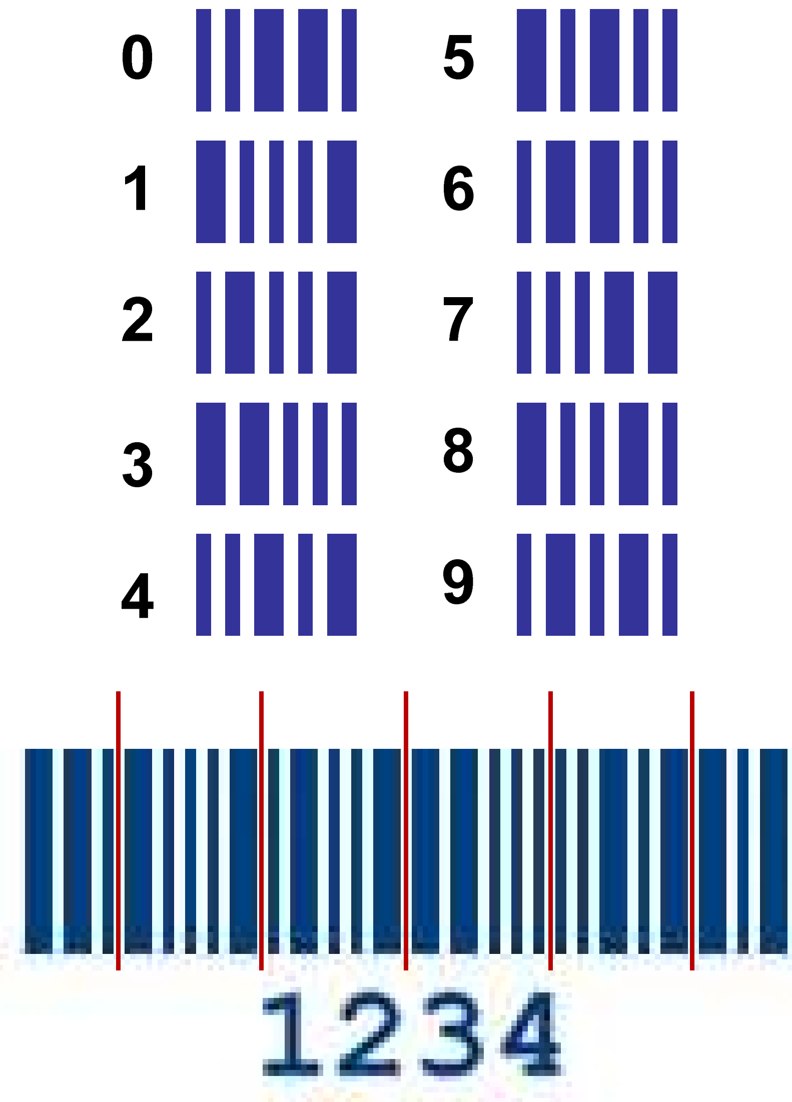
Kody liczbowe - Kod 2 z 5

- Kod 5-bitowy: 2 bity zawsze równe 1, a 3 bity zawsze równe 0
- Koduje 10 znaków (cyfry dziesiętne), kody nie są wzajemnie jednoznaczne (ta sama wartość może być zakodowana w różny sposób)
- Kod stałowagowy
- Kod detekcyjny
- Stosowany głównie w kodach kreskowych

Liczba dziesiętna	2 z 5 (01236)	2 z 5 (01234)	2 z 5 (74210)
0	01100	01100	11000
1	11000	11000	00011
2	10100	10100	00101
3	10010	10010	00110
4	01010	01010	01001
5	00110	00110	01010
6	10001	10001	01100
7	01001	01001	10001
8	00101	00101	10010
9	00011	00011	10100

Kody liczbowe - Kod 2 z 5 Industrial (1960 r.)

- Jednowymiarowy kod kreskowy kodujący cyfry: $0 \div 9$
- Znak to 5 pasków: 2 szerokie i 3 wąskie
- Szeroki pasek jest wielokrotnością wąskiego, szerokości muszą być takie same dla całego kodu
- Struktura kodu:
    - start: 11011010
    - numer
    - stop: 11010110



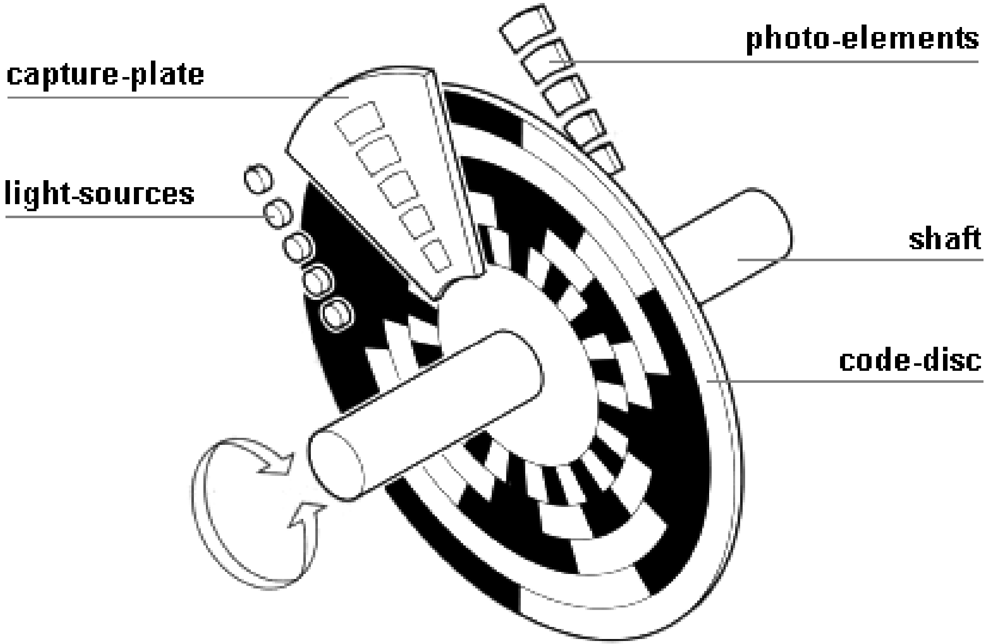
Kod Graya (refleksyjny)

- Kod dwójkowy, bezwagowy, niepozycyjny
- Dwa kolejne słowa kodowe różnią się stanem jednego bitu
- Kod cykliczny - ostatni i pierwszy wyraz również różnią się stanem jednego bitu

kod 1-bitowy	kod 2-bitowy	kod 3-bitowy
0	00	000
1	01	001
	11	011
	10	010
		110
		111
		101
		100

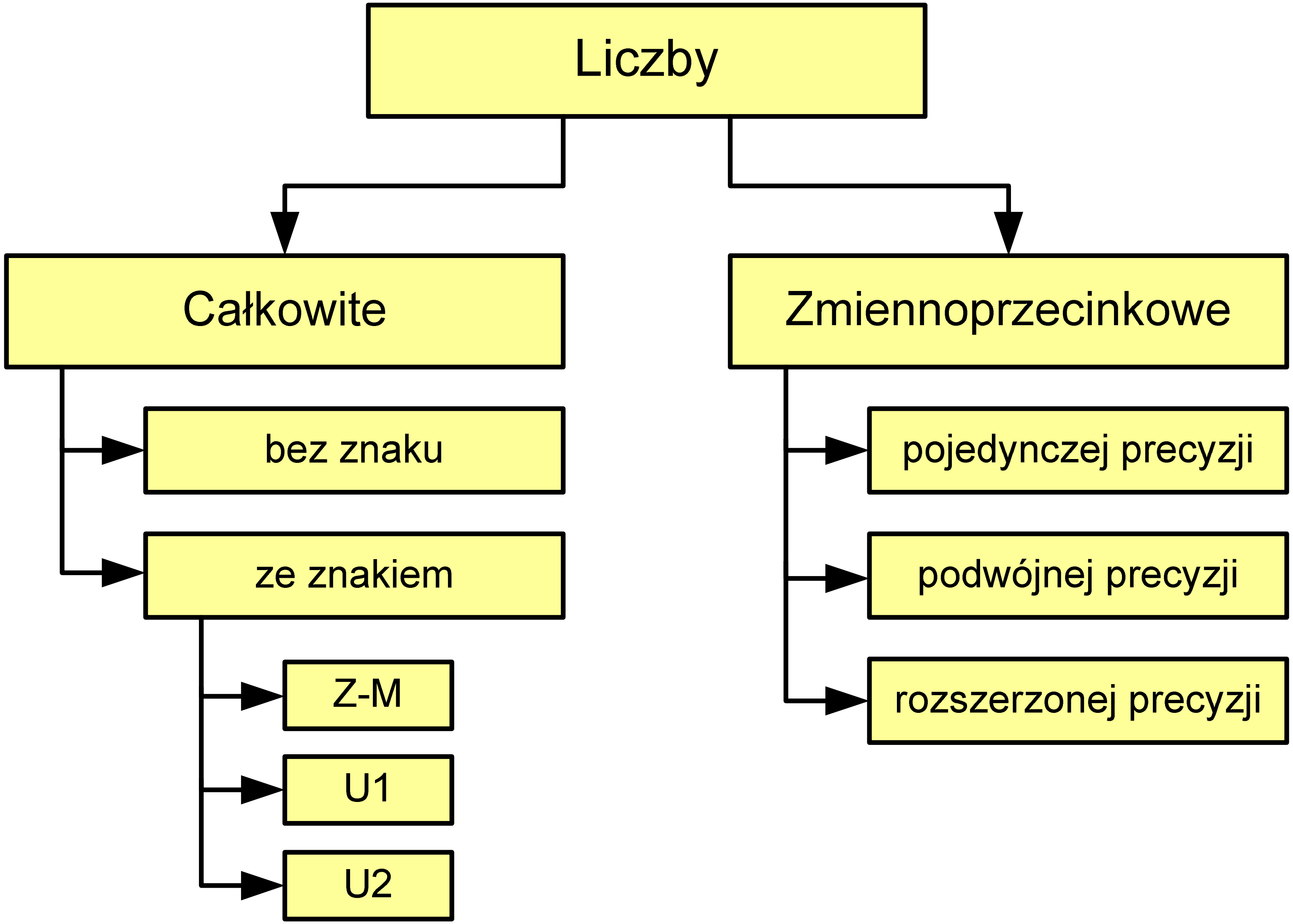
Kod Graya

- Stosowany w przetwornikach analogowo-cyfrowych, do cyfrowego pomiaru analogowych wielkości mechanicznych (np. kąt obrotu)



http://tams-www.informatik.uni-hamburg.de
/applets/hades/webdemos/10-gates/15-graycode/dual2gray.html

Reprezentacja liczb w systemach komputerowych



Liczby całkowite bez znaku

- Zapis liczby w systemie dwójkowym:

2^{n-1}	2^{n-2}		2^4	2^3	2^2	2^1	2^0	← **wagi**
x_{n-1}	x_{n-2}	...	x_4	x_3	x_2	x_1	x_0	← **cyfry (0 lub 1)**
n-1	n-2		4	3	2	1	0	← **pozycje**

n - bitów

- Używając n-bitów można zapisać liczbę z zakresu:

$$X_{(2)} = \left\langle 0,\ 2^n - 1 \right\rangle$$

$$8-bitów \qquad 0 \ldots 255$$
$$16-bitów \qquad 0 \ldots 65\,535$$
$$32-bity \qquad 0 \ldots 4\,294\,967\,295$$
$$64-bity \qquad 0 \ldots 18\,446\,744\,073\,709\,551\,615$$

18 trylionów 446 biliardów 744 biliony 73 miliardy 709 milionów 551 tysięcy 615

Liczby całkowite bez znaku w języku C

- Typy zmiennych całkowitych bez znaku stosowane w języku C:

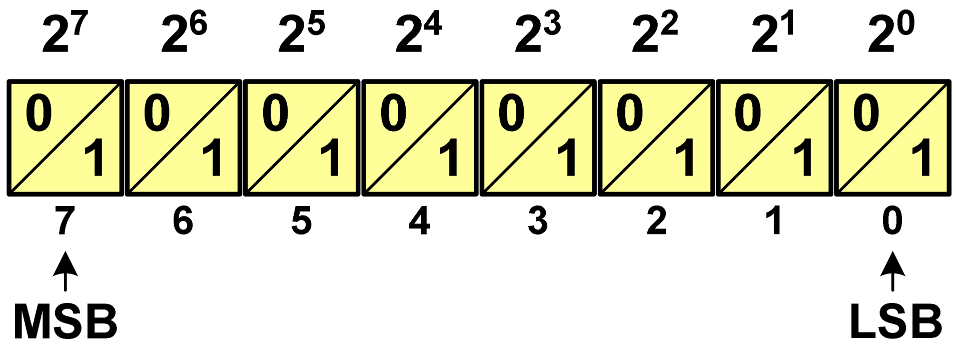
Nazwa typu	Rozmiar (bajty)	Zakres wartości
unsigned char	1 bajt	0 … 255
unsigned short int	2 bajty	0 … 65 535
unsigned int	4 bajty	0 … 4 294 967 295
unsigned long int	4 bajty	0 … 4 294 967 295
unsigned long long int	8 bajtów	0 … 18 446 744 073 709 551 615

- W nazwach typów short i long można pominąć słowo int:

unsigned short int	→	unsigned short
unsigned long int	→	unsigned long
unsigned long long int	→	unsigned long long

Liczby całkowite bez znaku w języku C

- Typ unsigned char (1 bajt):

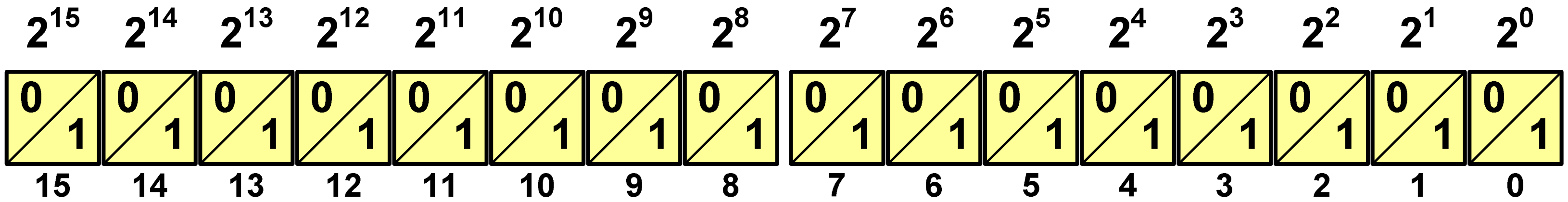


  - MSB (Most Significant Bit) - najbardziej znaczący bit, najstarszy bit, największa waga
  - LSB (Least Significant Bit) - najmniej znaczący bit, najmłodszy bit, najmniejsza waga

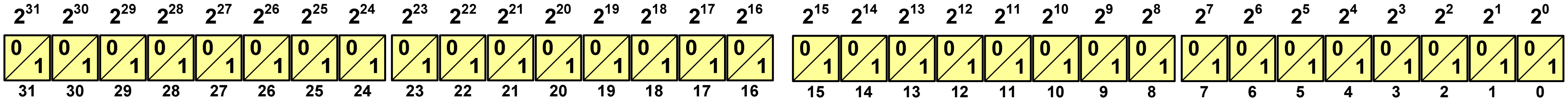
- Zakres wartości:
  - dolna granica: $0000\ 0000_{(2)} = 00_{(16)} = 0_{(10)}$
  - górna granica: $1111\ 1111_{(2)} = FF_{(16)} = 255_{(10)}$

Liczby całkowite bez znaku w języku C

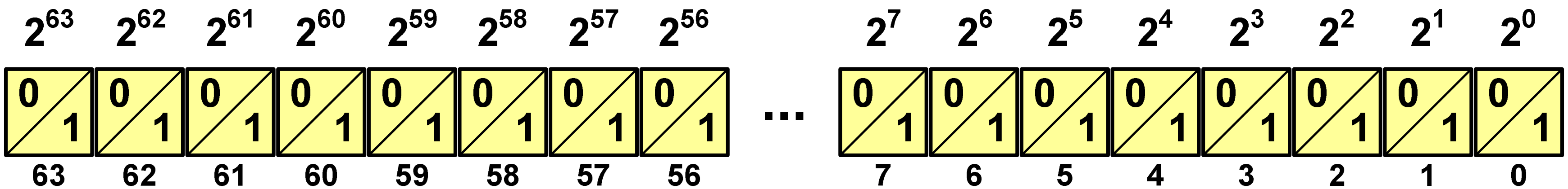
- Typ unsigned short int (2 bajty):



- Typy unsigned int (4 bajty) i unsigned long int (4 bajty):



- Typ unsigned long long int (8 bajtów):



Liczby całkowite bez znaku w języku C

```
unsigned short int:     65535 0 1
unsigned int:           4294967295 0 1
unsigned long int:      4294967295 0 1
unsigned long long int: 18446744073709551615 0 1
```

```
#include <stdi

int main()   /* przepełnienie zmiennej, ang. integer overflow */
{
   unsigned short int      usi = 65535;
   unsigned int             ui = 4294967295;
   unsigned long int       uli = 4294967295;
   unsigned long long int ulli = 18446744073709551615;

   printf("unsigned short int:     %hu %hu %hu\n",usi,usi+1,usi+2);
   printf("unsigned int:           %u %u %u\n",ui,ui+1,ui+2);
   printf("unsigned long int:      %lu %lu %lu\n",uli,uli+1,uli+2);
   printf("unsigned long long int: %llu %llu %llu\n",
                                    ulli,ulli+1,ulli+2);

   return 0;
}
```

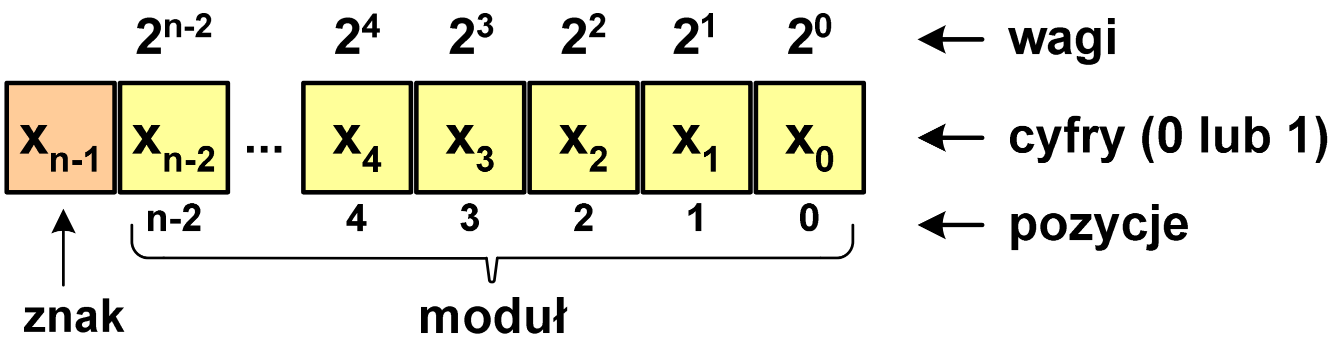
Liczby całkowite bez znaku w języku C

```
unsigned short int:     1 0 65535
unsigned int:           1 0 4294967295
unsigned long int:      1 0 4294967295
unsigned long long int: 1 0 18446744073709551615
```

```c
#include <stdi

int main()   /* przepełnienie zmiennej, ang. integer overflow */
{
   unsigned short int      usi = 1;
   unsigned int             ui = 1;
   unsigned long int       uli = 1;
   unsigned long long int ulli = 1;

   printf("unsigned short int:     %hu %hu %hu\n",usi,usi-1,usi-2);
   printf("unsigned int:           %u %u %u\n",ui,ui-1,ui-2);
   printf("unsigned long int:      %lu %lu %lu\n",uli,uli-1,uli-2);
   printf("unsigned long long int: %llu %llu %llu\n",
                                    ulli,ulli-1,ulli-2);

   return 0;
}
```

Liczby całkowite ze znakiem - kod znak-moduł

- Inne nazwy: ZM, Z-M, SM (Signed Magnitude), S+M
- Najstarszy bit jest bitem znaku liczby: 0 - dodatnia, 1 - ujemna
- Pozostałe bity mają takie same znaczenie jak w NKB



- Wartość liczby:

$$X_{(10)} = \underbrace{(x_0 \cdot 2^0 + x_1 \cdot 2^1 + x_2 \cdot 2^2 + \ldots + x_{n-2} \cdot 2^{n-2})}_{\text{moduł}} \cdot \underbrace{(-1)^{x_{n-1}}}_{\text{znak}} = (-1)^{x_{n-1}} \cdot \sum_{i=0}^{n-2} x_i \cdot 2^i$$

Liczby całkowite ze znakiem - kod znak-moduł

- Liczby 4-bitowe (1 bit - znak, 3 bity - moduł) w kodzie Z-M:

Z-M	dziesiętnie
0000	+0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Z-M	dziesiętnie
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

- dwie reprezentacje zera

$$+0\ (0000_{ZM})$$

$$-0\ (1000_{ZM})$$

- Zakres liczb dla n-bitów:

$$X_{(10)} = \left\langle -2^{n-1}+1,\ 2^{n-1}-1 \right\rangle$$

$$\text{dla 8 bitów}: \quad \langle -127 \ldots 127 \rangle$$

$$\text{dla 16 bitów}: \quad \langle -32767 \ldots 32767 \rangle$$

Liczby całkowite ze znakiem - kod znak-moduł

- Zamiana liczby dziesiętnej na kod Z-M:
  - liczba dodatnia

$$93_{(10)} = ?_{(ZM)}$$

  - zamieniamy liczbę na NKB

$$93_{(10)} = 1011101_{(NKB)}$$

  - dodajemy bit znaku

$$93_{(10)} = \boxed{0}1011101_{(ZM)}$$

  - liczba ujemna

$$-93_{(10)} = ?_{(ZM)}$$

  - zamieniamy moduł liczby na NKB

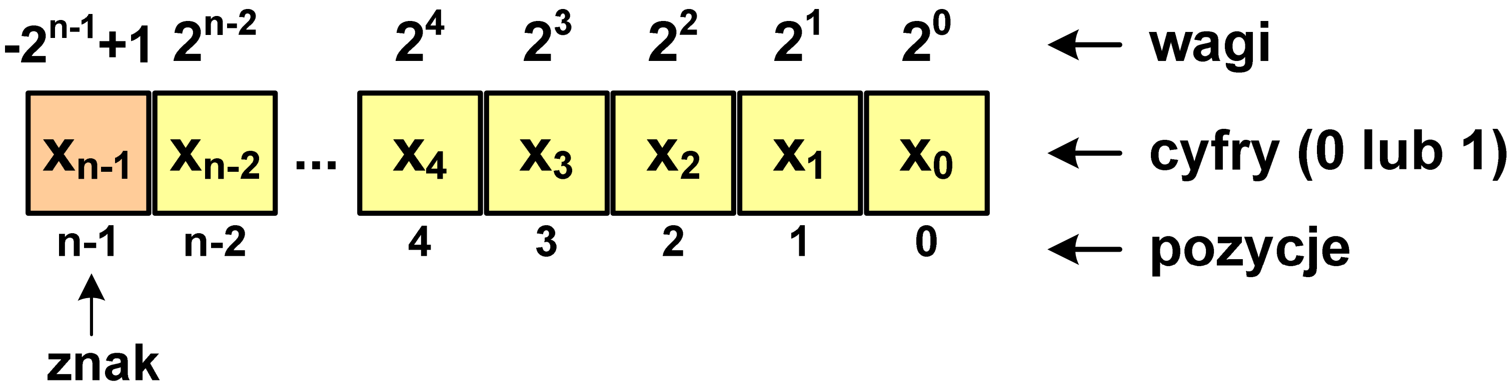
$$\left|-93_{(10)}\right| = 93_{(10)} = 1011101_{(NKB)}$$

  - dodajemy bit znaku

$$-93_{(10)} = \boxed{1}1011101_{(ZM)}$$

Liczby całkowite ze znakiem - kod U1

- Inne nazwy: U1, ZU1, uzupełnień do jedności
- Najstarszy bit jest bitem znaku liczby: 0 - dodatnia, 1 - ujemna
- Wszystkie bity liczby posiadają takie same wagi jak w NKB, oprócz pierwszego bitu, który ma wagę $-2^{n-1} + 1$



- Wartość liczby:

$$X_{(10)} = x_0 \cdot 2^0 + x_1 \cdot 2^1 + x_2 \cdot 2^2 + \ldots + x_{n-2} \cdot 2^{n-2} + x_{n-1} \cdot (-2^{n-1} + 1)$$

Liczby całkowite ze znakiem - kod U1

- Liczby 4-bitowe (1 bit - znak, 3 bity - moduł) w kodzie U1:

U1	dziesiętnie
0000	+0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

U1	dziesiętnie
1111	-0
1110	-1
1101	-2
1100	-3
1011	-4
1010	-5
1001	-6
1000	-7

- liczby dodatnie zapisywane są tak samo jak w NKB
- liczby ujemne otrzymywane są poprzez bitową negację
- dwie reprezentacje zera

- Zakres liczb dla n-bitów:

$$X_{(10)} = \left\langle -2^{n-1}+1,\ 2^{n-1}-1 \right\rangle$$

$$\text{dla 8 bitów}: \quad \langle -127 \ldots 127 \rangle$$

$$\text{dla 16 bitów}: \quad \langle -32767 \ldots 32767 \rangle$$

Liczby całkowite ze znakiem - kod U1

- Zamiana liczby dziesiętnej na kod U1:

  - liczba dodatnia

$$93_{(10)} = ?_{(U1)}$$

  - zamieniamy liczbę na NKB

$$93_{(10)} = 1011101_{(NKB)}$$

  - dodajemy bit znaku: 0

$$93_{(10)} = 01011101_{(U1)}$$

  - liczba ujemna

$$-93_{(10)} = ?_{(U1)}$$

  - zamieniamy moduł liczby na U1

$$\left|-93_{(10)}\right| = 93_{(10)} = 01011101_{(U1)}$$

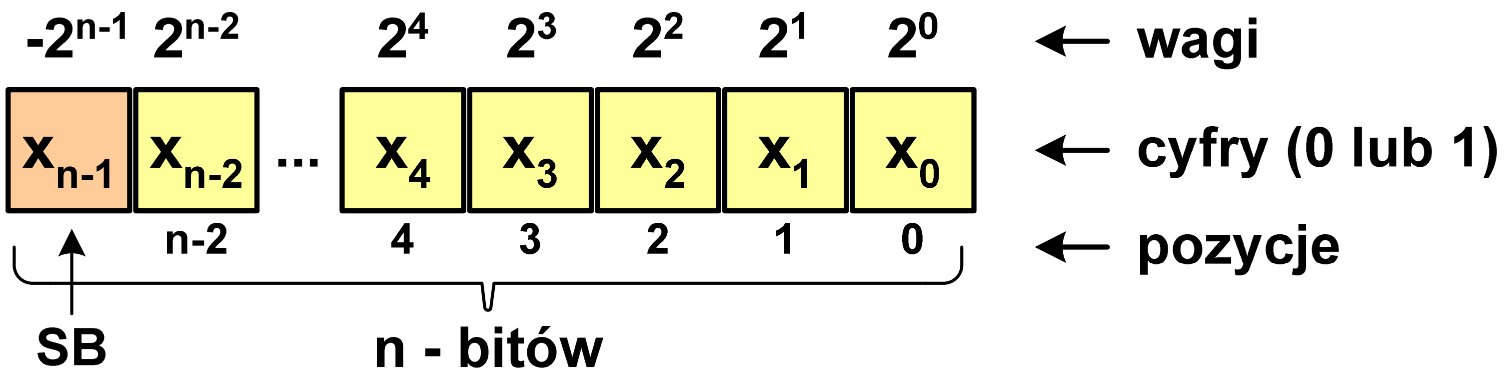
  - negujemy wszystkie bity

$$-93_{(10)} = 10100010_{(U1)}$$

bit znaku

Liczby całkowite ze znakiem - kod U2

- Inne nazwy: ZU2, uzupełnień do dwóch, two's complement
- Najstarszy bit jest bitem znaku liczby: 0 - dodatnia, 1 - ujemna



- Wartość liczby:

$$X_{(10)} = x_0 \cdot 2^0 + x_1 \cdot 2^1 + x_2 \cdot 2^2 + \ldots + x_{n-2} \cdot 2^{n-2} + x_{n-1} \cdot (-2^{n-1})$$

- Kod U2 jest obecnie powszechnie stosowany w informatyce

Liczby całkowite ze znakiem - kod U2

- Liczby 4-bitowe (1 bit - znak, 3 bity - moduł) w kodzie U2:

U2	dziesiętnie	U2	dziesiętnie
0000	0	1111	-1
0001	1	1110	-2
0010	2	1101	-3
0011	3	1100	-4
0100	4	1011	-5
0101	5	1010	-6
0110	6	1001	-7
0111	7	1000	-8

- brak podwójnej reprezentacji zera
- liczb ujemnych jest o jeden więcej niż dodatnich
- 00...000 zawsze oznacza $0_{(10)}$
  11...111 zawsze oznacza $-1_{(10)}$

- Zakres liczb dla n-bitów:

$$X_{(10)} = \left\langle -2^{n-1},\ 2^{n-1}-1 \right\rangle$$

$$\text{dla 8 bitów}: \quad \left\langle -128 \ \ldots\ 127 \right\rangle$$

$$\text{dla 16 bitów}: \quad \left\langle -32768 \ \ldots\ 32767 \right\rangle$$

Liczby całkowite ze znakiem - kod U2

- Zamiana liczby dziesiętnej na kod U2:

  - liczba dodatnia

$$75_{(10)} = ?_{(U2)}$$

  - zamieniamy liczbę na NKB

$$75_{(10)} = 1001011_{(NKB)}$$

  - dodajemy bit znaku: 0

$$75_{(10)} = \boxed{0}1001011_{(U2)}$$

  - liczba ujemna

$$-75_{(10)} = ?_{(U2)}$$

  - zamieniamy moduł liczby na U2

$$\left|-75_{(10)}\right| = 75_{(10)} = 01001011_{(U2)}$$

  - negujemy wszystkie bity i dodajemy 1

$$\begin{array}{rr} & 01001011 \\ \text{negacja:} & 10110100 \\ +1: & 1 \\ \hline \end{array}$$

$$-75_{(10)} = 10110101_{(U2)}$$

Liczby całkowite ze znakiem - kod U2 w języku C

- Typy zmiennych całkowitych ze znakiem stosowane w języku C:

Nazwa typu	Rozmiar (bajty)	Zakres wartości
char	1 bajt	-128 ... 127
short int	2 bajty	-32 768 ... 32 767
int	4 bajty	-2 147 483 648 ... 2 147 483 647
long int	4 bajty	-2 147 483 648 ... 2 147 483 647
long long int	8 bajtów	-9 223 372 036 854 775 808 ... 9 223 372 036 854 775 807

- Przed nazwą każdego z powyższych typów można dodać signed

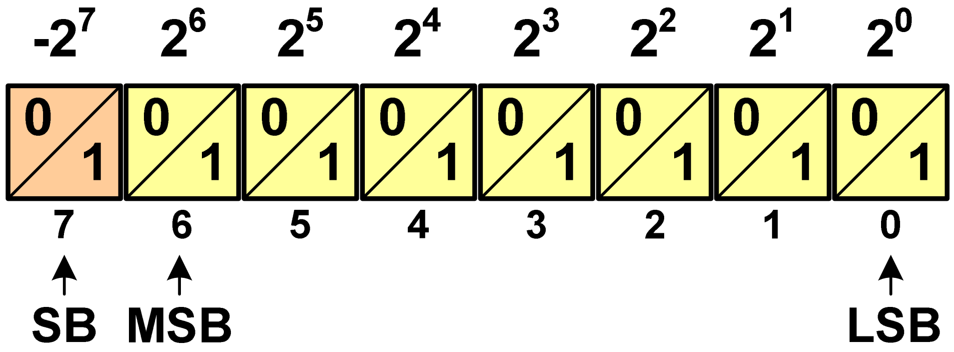
  signed char, signed short int, signed int ...

- W nazwach typów short i long można pominąć słowo int:

  short int → short, long int → long, long long int → long long

Liczby całkowite ze znakiem - kod U2 w języku C

- Typ char / signed char (1 bajt):

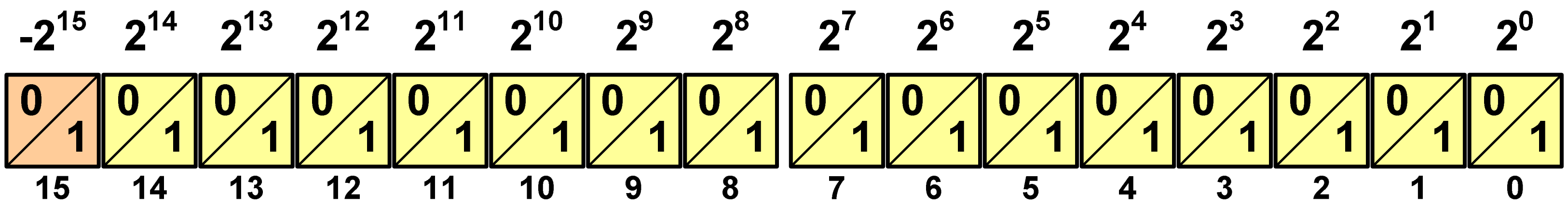


- Zakres wartości:
  - dolna granica: $1000\,0000_{(2)} = -128_{(10)}$
  - górna granica: $0111\,1111_{(2)} = 127_{(10)}$
  - inne wartości: $1111\,1111_{(2)} = -1_{(10)}$

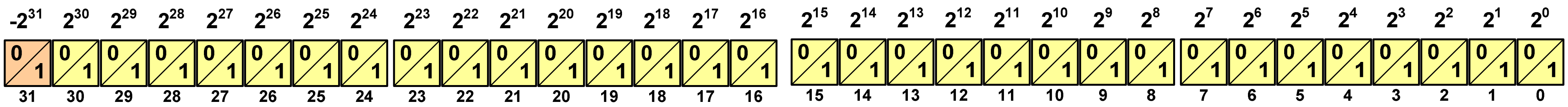
    $0000\,0000_{(2)} = 0_{(10)}$

Liczby całkowite bez znaku w języku C

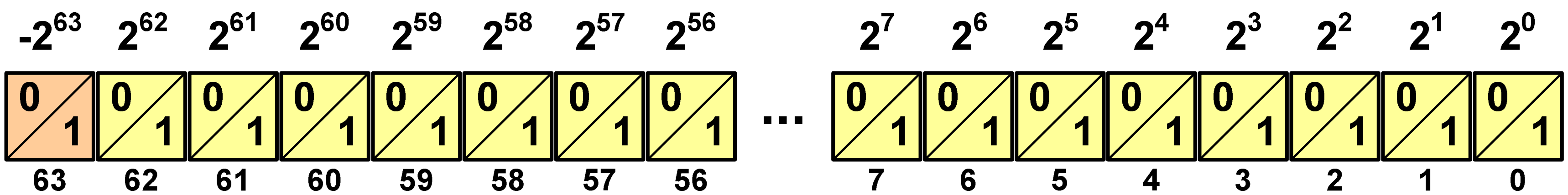
- Typ short / signed short int (2 bajty):



- Typy int / signed int (4 bajty) i long / signed long int (4 bajty):



- Typ long long int / signed long long int (8 bajtów):



Liczby całkowite ze znakiem - kod U2 w języku C

```
short int:     32767 -32768 -32767
int:           2147483647 -2147483648 -2147483647
long int:      2147483647 -2147483648 -2147483647
long long int: 9223372036854775807 -9223372036854775808
```

```c
#include

int main()   /* przepełnienie zmiennej, ang. integer overflow */
{
    short int      si = 32767;
    int             i = 2147483647;
    long int       li = 2147483647;
    long long int lli = 9223372036854775807;

    printf("short int:     %hd %hd %hd\n",si,si+1,si+2);
    printf("int:           %d %d %d\n",i,i+1,i+2);
    printf("long int:      %ld %ld %ld\n",li,li+1,li+2);
    printf("long long int: %lld %lld\n",lli,lli+1);

    return 0;
}
```

Zapis zmiennoprzecinkowy liczby rzeczywistej

- Zapis bardzo dużych lub małych liczb wymaga dużej liczby cyfr
- Znacznie prostsze jest przedstawienie liczb w postaci zmiennoprzecinkowej (ang. floating point numbers)
  - 12 000 000 000 000 = $1{,}2 \cdot 10^{13}$
  - 0,000 000 000 001 = $1{,}0 \cdot 10^{-12}$
- Zapis liczby zmiennoprzecinkowej ma postać:

$$L = M \cdot B^{E}$$

gdzie:

L - wartość liczby

M - mantysa

B - podstawa systemu

E - wykładnik, cecha

- notacja naukowa: 1,2e13 1,2e+13 1,2E13 1,2E+13
- postać wykładnicza: $1{,}2 \cdot 10^{13}$

Zapis zmiennoprzecinkowy liczby rzeczywistej

$$2{,}43\cdot 10^{3}{}_{(10)} = 2{,}43\cdot 1000 = 2430_{(10)}$$

$$6{,}59\cdot 10^{-2}{}_{(10)} = 6{,}59\cdot 0{,}01 = 0{,}0659_{(10)}$$

$$1{,}011\cdot 10^{101}{}_{(2)} = ?_{(10)}$$

$$M = 1{,}011_{(2)} = 1\cdot 2^{0} + 0\cdot 2^{-1} + 1\cdot 2^{-2} + 1\cdot 2^{-3} = 1{,}375_{(10)}$$

$$B = 10_{(2)} = 0\cdot 2^{0} + 1\cdot 2^{1} = 2_{(10)}$$

$$E = 101_{(2)} = 1\cdot 2^{0} + 0\cdot 2^{1} + 1\cdot 2^{2} = 1 + 4 = 5_{(10)}$$

$$1{,}011\cdot 10^{101}{}_{(2)} = 1{,}375\cdot 2^{5} = 1{,}375\cdot 32 = 44_{(10)}$$

$$3{,}121\cdot 10^{32}{}_{(4)} = ?_{(10)}$$

$$M = 3{,}121_{(4)} = 3\cdot 4^{0} + 1\cdot 4^{-1} + 2\cdot 4^{-2} + 1\cdot 4^{-3} = 3{,}390625_{(10)}$$

$$B = 10_{(4)} = 0\cdot 4^{0} + 1\cdot 4^{1} = 4_{(10)}$$

$$E = 32_{(4)} = 2\cdot 4^{0} + 3\cdot 4^{1} = 2 + 12 = 14_{(10)}$$

$$3{,}121\cdot 10^{32}{}_{(4)} = 3{,}390625\cdot 4^{14} = 910\,163\,968_{(10)}$$

Postać znormalizowana zapisu liczby

- Położenie przecinka w mantysie nie jest ustalone i może się zmieniać
- Poniższe zapisy oznaczają tę samą liczbę (system dziesiętny)

$$243 \cdot 10^1 = 24{,}3 \cdot 10^2 = 2{,}43 \cdot 10^3 = 0{,}243 \cdot 10^4$$

- Dla ujednolicenia zapisu i usunięcia wielokrotnych reprezentacji tej samej liczby, przyjęto tzw. postać znormalizowaną zapisu liczby
- W postaci znormalizowanej mantysa spełnia nierówność:

$$B > |M| \geq 1$$

Przykład:

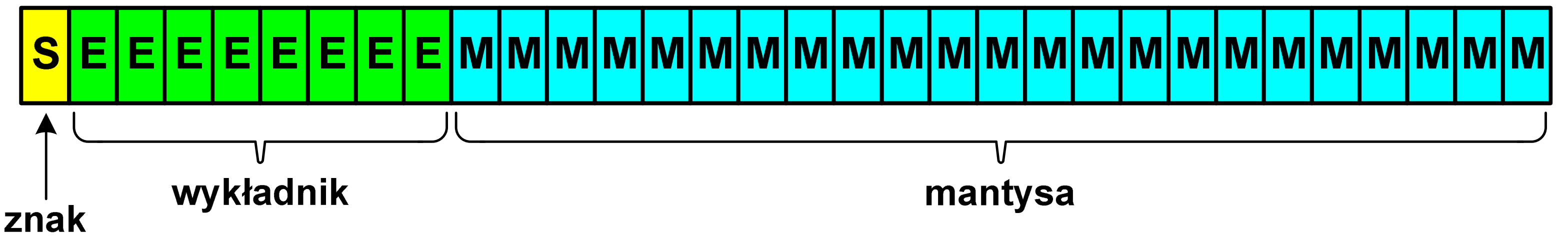
$2{,}43 \cdot 10^3$ - to jest postać znormalizowana, gdyż: $10 > |2{,}43| \geq 1$

$0{,}243 \cdot 10^4$ - to nie jest postać znormalizowana

$24{,}3 \cdot 10^2$ - to nie jest postać znormalizowana

Liczby zmiennoprzecinkowe w systemie binarnym

- Liczba bitów przeznaczonych na mantysę i wykładnik jest ograniczona



- Wartość liczby L:

$$L = (-1)^{S} \cdot M \cdot B^{E}$$

gdzie:

S - znak liczby (ang. sign), przyjmuje wartość 0 lub 1
M - znormalizowana mantysa (ang. mantissa), liczba ułamkowa
B - podstawa systemu liczbowego (ang. base)
E - wykładnik (ang. exponent), cecha, liczba całkowita

- W systemie binarnym podstawa systemu jest stała: B = 2

$$L = (-1)^{S} \cdot M \cdot 2^{E}$$

Przesunięcie wykładnika

- Wykładnik zapisywany jest z przesunięciem (ang. bias)

$$L = (-1)^{S} \cdot M \cdot 2^{E-BIAS}$$

gdzie:

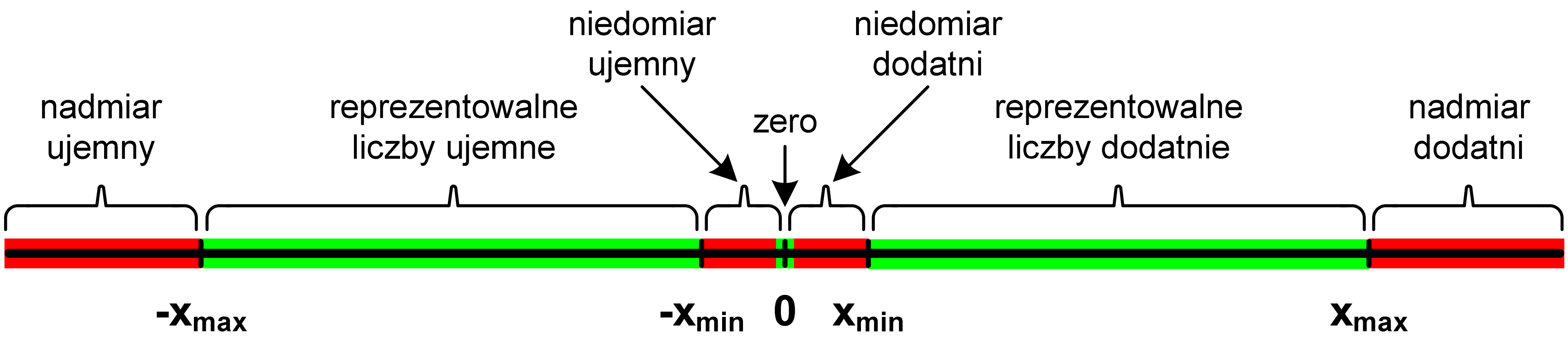
L - wartość liczby   S - znak liczby   M - mantysa
E - wykładnik   BIAS - przesunięcie (nadmiar)

- Typowe wartości przesunięcia (nadmiaru) wynoszą:
  - formatu 32-bitowy: 2^{7}-1 = $127_{(10)}$ = $7F_{(16)}$
  - formatu 64-bitowy: 2^{10}-1 = $1023_{(10)}$ = $3FF_{(16)}$
  - formatu 80-bitowy: 2^{14}-1 = $16383_{(10)}$ = $3FFF_{(16)}$

Zakres liczb zmiennoprzecinkowych

- Zakres liczb w zapisie zmiennoprzecinkowym:

$$\langle -x_{max}, -x_{min} \rangle \cup \{0\} \cup \langle x_{min}, x_{max} \rangle$$



- Największa i najmniejsza wartość liczby w danej reprezentacji:

$$x_{min} = M_{min} \cdot B^{E_{min}} \qquad x_{max} = M_{max} \cdot B^{E_{max}}$$

Koniec wykładu nr 5

Dziękuję za uwagę!