Informatyka 1 (ES1F1002)

Politechnika Białostocka - Wydział Elektryczny

Elektrotechnika, semestr II, studia stacjonarne I stopnia

Rok akademicki 2022/2023

Wykład nr 4 (24.10.2022)

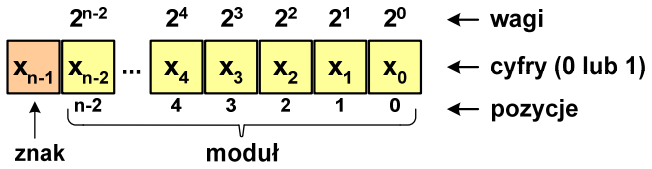
dr inż. Jarosław Forenc

Plan wykładu nr 4

- Reprezentacja liczb całkowitych
  - liczby ze znakiem (ZM, U1, U2)
- Język C
  - instrukcja if, operatory relacyjne i logiczne, wyrażenia logiczne
  - operator warunkowy, instrukcja switch
- Reprezentacja zmiennoprzecinkowa
  - zapis, postać znormalizowana
  - zakres liczb zmiennoprzecinkowych
- Standard IEEE 754
  - liczby 32-bitowe, liczby 64-bitowe
  - zakres i precyzja liczb
  - wartości specjalne

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

- dwie reprezentacje zera

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

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

- Zakres liczb dla n-bitów:

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

dla 8 bitów: $\langle -127 \ldots 127 \rangle$

dla 16 bitów: $\langle -32767 \ldots 32767 \rangle$

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

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

  - liczba ujemna

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

  - zamieniamy moduł liczby na NKB

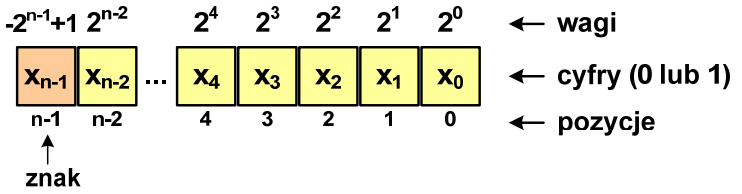
$$\left|-93_{(10)}\right| = 93_{(10)} = 1011101_{(NKB)}$$

  - dodajemy bit znaku

$$-93_{(10)} = 11011101_{(ZM)}$$

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

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

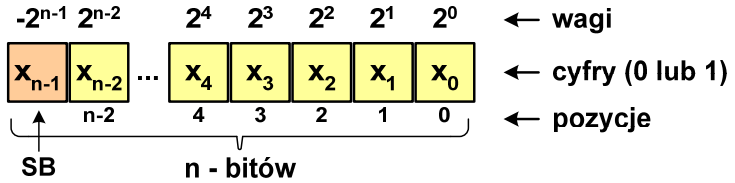
  - negujemy wszystkie bity

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

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

U2	dziesiętnie
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-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:

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

dla 8 bitów: $\langle -128 \ldots 127 \rangle$

dla 16 bitów: $\langle -32768 \ldots 32767 \rangle$

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

$$75_{(10)} = 01001011_{(U2)}$$

  - liczba ujemna

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

  - zamieniamy moduł liczby na U2

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

  - negujemy wszystkie bity i dodajemy 1

$$\begin{array}{rr} & 01001011 \\ \text{negacja:} & 10110100 \\ +1: & 1 \\ \hline -75_{(10)} = & 10110101_{(U2)} \end{array}$$

Liczby całkowite ze znakiem - kod U2 w języku C

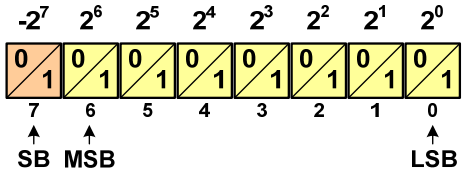
- Typy zmiennych całkowitych ze znakiem stosowane w języku C:

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

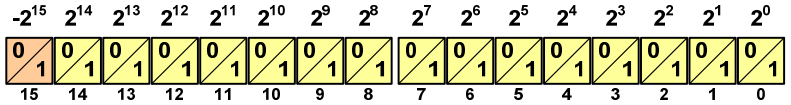
- Typ char / signed char (1 bajt):



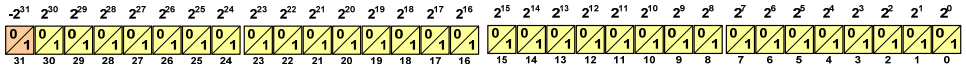
- Zakres wartości:
  - dolna granica: $1000\ 0000_{(2)} = -128_{(10)}$
  - górna granica: $0111\ 1111_{(2)} = 127_{(10)}$
  - inne wartości: $1111\ 1111_{(2)} = -1_{(10)}$
    $0000\ 0000_{(2)} = 0_{(10)}$

Liczby całkowite bez znaku w języku C

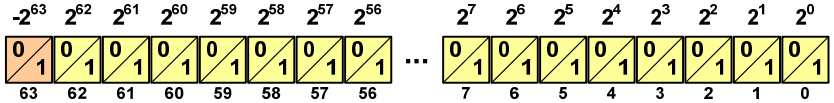
- Typ short / signed short int (2 bajty):



- 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
```

```
#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;
}
```


Przykład: pierwiastek kwadratowy

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

int main(void)
{
    float x, y;

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

    y = sqrt(x);

    printf("Pierwiastek liczby: %f\n",y);

    return 0;
}
```

```
Podaj liczbe: 15
Pierwiastek liczby: 3.872983
```

```
Podaj liczbe: -15
Pierwiastek liczby: -1.#IND00
```

Przykład: 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: 15
Pierwiastek liczby: 3.872983
```

```
Podaj liczbe: -15
Blad! Liczba ujemna
```


Język C - instrukcja warunkowa if

```
if (wyrażenie)
    instrukcja1
```

- jeśli wyrażenie jest prawdziwe, to wykonywana jest instrukcja1
- gdy wyrażenie jest fałszywe, to instrukcja1 nie jest wykonywana

```
if (wyrażenie)
    instrukcja1
else
    instrukcja2
```

- jeśli wyrażenie jest prawdziwe, to wykonywana jest instrukcja1, zaś instrukcja2 nie jest wykonywana
- gdy wyrażenie jest fałszywe, to wykonywana jest instrukcja2, zaś instrukcja1 nie jest wykonywana

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

Język C - instrukcja warunkowa if

```
if (wyrażenie)
    instrukcja
```

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

```
if (x>0)
    printf("inst1");
```

```
if (x>0)
{
    printf("inst1");
    printf("inst2");
    ...
}
```


Język C - instrukcja warunkowa if

```
if (wyr)
    instr;
```

```
if (wyr)
{
    instr;
    instr;
}
```

```
if (wyr)
    instr;
else
    instr;
```

```
if (wyr)
{
    instr;
    instr;
}
else
{
    instr;
    instr;
}
```

```
if (wyr)
{
    instr;
    instr;
}
else
    instr;
```

```
if (wyr)
    instr;
else
{
    instr;
    instr;
}
```

```
if (wyr)
{
    instr;
}
else
{
    instr;
}
```

Język C - Operatory relacyjne (porównania)

Operator	Przykład	Znaczenie
`>`	`a > b`	a większe od b
`<`	`a < b`	a mniejsze od b
`>=`	`a >= b`	a większe lub równe b
`<=`	`a <= b`	a mniejsze lub równe b
`==`	`a == b`	a równe b
`!=`	`a != b`	a nierówne b (a różne od b)

- Wynik porównania jest wartością typu int i jest równy:
  - 1 - gdy warunek jest prawdziwy
  - 0 - gdy warunek jest fałszywy (nie jest prawdziwy)

Język C - Operatory logiczne

Operator	Znaczenie	Opis
`!`	`NOT, nie`	jednoargumentowy operator negacji logicznej - zmienia argument różny od zera na wartość 0, a argument równy zero na wartość 1
`&&`	`AND, i`	dwuargumentowy operator koniunkcji, iloczyn logiczny
`||`	`OR, lub`	dwuargumentowy operator alternatywy, suma logiczna

- Wynikiem zastosowania operatorów logicznych && i || jest wartość typu int równa 1 (prawda) lub 0 (fałsz)

```
if (x>5 && x<8)
```

```
if (x<=5 || x>8)
```


Język C - Wyrażenia logiczne

- Wyrażenia logiczne mogą zawierać:
  - operatory relacyjne
  - operatory logiczne
  - operatory arytmetyczne
  - operatory przypisania
  - zmienne
  - stałe
  - wywołania funkcji
  - ...
- Kolejność operacji wynika z priorytetu operatorów

Operator	Typ operatora
`!`	logiczny
`* / %`	arytmetyczne
`+ -`	arytmetyczne
`> < >= <=`	relacyjne
`== !=`	relacyjne
`&&`	logiczny
`||`	logiczny
`=`	przypisania

Język C - Wyrażenia logiczne

```
int  x = 0, y = 1, z = 2;
```

```
if ( x == 0 )
```
wynik: 1 (prawda)

```
if ( x = 0 )
```
wynik: 0 (fałsz) (!!!)

```
if ( x != 0 )
```
wynik: 0 (fałsz)

```
if ( x =! 0 )
```
wynik: 1 (prawda) (!!!)

```
if ( z > x + y )
```

```
if ( z > (x + y) )
```
wynik: 1 (prawda)

Język C - Wyrażenia logiczne

```
int  x = 0, y = 1, z = 2;
```

```
if ( x>2 && x<5 )
```

```
if ( (x>2) && (x<5) )
```

wynik: 0 (fałsz)

- Wyrażenia logiczne obliczane są od strony lewej do prawej
- Proces obliczeń kończy się, gdy wiadomo, jaki będzie wynik całego wyrażenia

```
if ( 2 < x < 5 )
```

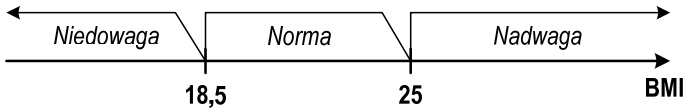
wynik: 1 (prawda) (!!!)

Przykład: obliczanie BMI (Body Mass Index)

- BMI - współczynnik powstały przez podzielenie masy ciała podanej w kilogramach przez kwadrat wzrostu podanego w metrach

$$BMI = \frac{masa}{wzrost^2}$$

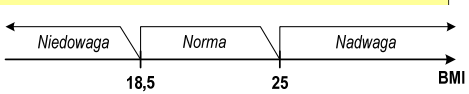
- Dla osób dorosłych:
  - BMI < 18,5 - wskazuje na niedowagę
  - BMI ≥ 18,5 i BMI < 25 - wskazuje na prawidłową masę ciała
  - BMI ≥ 25 - wskazuje na nadwagę



Przykład: obliczanie BMI (Body Mass Index)

```
#include <stdio.h>

int main(void)
{
    double masa, wzrost, bmi;

    printf("Podaj mase [kg]:  "); scanf("%lf",&masa);
    printf("Podaj wzrost [m]: "); scanf("%lf",&wzrost);
    bmi = masa / (wzrost*wzrost);
    printf("bmi: %.2f\n",bmi);

    if (bmi<18.5)
        printf("Niedowaga\n");
    if (bmi>=18.5 && bmi<25)
        printf("Norma\n");
    if (bmi>=25)
        printf("Nadwaga\n");

    return 0;
}
```

```
Podaj mase [kg]:  84
Podaj wzrost [m]: 1.85
bmi: 24.54
Norma
```



Przykład: obliczanie BMI (Body Mass Index)

- Zamiast trzech instrukcji if:

```
if (bmi<18.5)
    printf("Niedowaga\n");
if (bmi>=18.5 && bmi<25)
    printf("Norma\n");
if (bmi>=25)
    printf("Nadwaga\n");
```

można zastosować tylko dwie:

```
if (bmi<18.5)
    printf("Niedowaga\n");
else
    if (bmi<25)
        printf("Norma\n");
    else
        printf("Nadwaga\n");
```

Język C - Operator warunkowy

- Operator warunkowy skład się z dwóch symboli i trzech operandów

```
wyrażenie1 ? wyrażenie2 : wyrażenie3
```

- Najczęściej zastępuje proste instrukcje if-else

```
float akcyza, cena, pojemnosc;
```

```
if (pojemnosc <= 2000)
    akcyza = cena*0.031;    /*  3.1% */
else
    akcyza = cena*0.186;    /* 18.6% */
```

```
akcyza = pojemnosc <= 2000 ? cena*0.031 : cena*0.186;
```


Język C - Operator warunkowy

```
if (x < 0)
    y = -x;
else
    y = x;
```

```
y = x < 0 ? -x : x;
```

- obliczenie modułu liczby x

```
if (a > b)
    max = a;
else
    max = b;
```

```
max = a > b ? a : b;
```

- wyznaczenie max z dwóch liczb

- Operator warunkowy ma bardzo niski priorytet
- Niższy priorytet mają tylko operatory przypisania (=, +=, -=,...) i operator przecinkowy (,)

Przykład: operator warunkowy

- Studenci chcą dojechać z akademika do sklepu - ile taksówek powinni zamówić? (Jedna taksówka może przewieźć 4 osoby.)

```
#include <stdio.h>

int main(void)
{
    int st, taxi;

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

    taxi = st / 4 + (st % 4 != 0 ? 1 : 0);

    printf("Liczba taxi: %d\n",taxi);

    return 0;
}
```

```
Podaj liczbe studentow: 23
Liczba taxi: 6
```


Przykład: sprawdzenie parzystości liczby

```
#include <stdio.h>

int main(void)
{
   int x;

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

   if (x%2==0)
       printf("Liczba parzysta\n");
   else
       printf("Liczba nieparzysta\n");

   printf("Liczba %s\n",x%2==0 ? "parzysta":"nieparzysta");

   return 0;
}
```

```
Podaj x: -3
Liczba nieparzysta
Liczba nieparzysta
```

Język C - Instrukcja switch

- Instrukcja wyboru wielowariantowego switch

```
switch (wyrażenie)
{
    case wyrażenie_stałe: instrukcje;
    case wyrażenie_stałe: instrukcje;
    case wyrażenie_stałe: instrukcje;
    ...
    default: instrukcje;
}
```

- wyrażenie_stałe - wartość typu całkowitego, znana podczas kompilacji
  - stała liczbowa, np. 3, 5, 9
  - znak w apostrofach, np. 'a', 'z', '+'
  - stała zdefiniowana przez const lub #define

Język C - Instrukcja switch

- Program wyświetlający słownie liczbę z zakresu 1..5 wprowadzoną z klawiatury

```
#include <stdio.h>

int main(void)
{
     int liczba;

     printf("Podaj liczbe (1..5): ");
     scanf("%d",&liczba);
```


Język C - Instrukcja switch

```
switch (liczba)
{
   case 1: printf("Liczba: jeden\n");
           break;
   case 2: printf("Liczba: dwa\n");
           break;
   case 3: printf("Liczba: trzy\n");
           break;
   case 4: printf("Liczba: cztery\n");
           break;
   case 5: printf("Liczba: piec\n");
           break;
   default: printf("Inna liczba\n");
}
```

```
Podaj liczbe: 2
Liczba: dwa
```

```
Podaj liczbe: 0
Inna liczba
```


Język C - Instrukcja switch

```
switch (liczba)
{
   case 1:
   case 3:
   case 5: printf("Liczba nieparzysta\n");
           break;
   case 2:
   case 4: printf("Liczba parzysta\n");
           break;
   default: printf("Inna liczba\n");
}
```

```
Podaj liczbe: 2
Liczba parzysta
```

- Te same instrukcje mogą być wykonane dla kilku etykiet case

Język C - Instrukcja switch

```
switch (liczba)
{
   case 1: case 3: case 5:
           printf("Liczba nieparzysta\n");
           break;
   case 2: case 4:
           printf("Liczba parzysta\n");
           break;
   default: printf("Inna liczba\n");
}
```

```
Podaj liczbe: 2
Liczba parzysta
```

- Etykiety case mogą być pisane w jednym wierszu

Język C - Instrukcja switch

```
switch (liczba%2)
{
   case 1: case -1:
           printf("Liczba nieparzysta\n");
           break;
   case 0:
           printf("Liczba parzysta\n");
}
```

```
Podaj liczbe: 2
Liczba parzysta
```

- Część domyślna (default) może być pominięta

Język C - Instrukcja switch (bez break)

```
switch (liczba)
{
   case 1: printf("Liczba: jeden\n");
   case 2: printf("Liczba: dwa\n");
   case 3: printf("Liczba: trzy\n");
   case 4: printf("Liczba: cztery\n");
   case 5: printf("Liczba: piec\n");
   default: printf("Inna liczba\n");
}
```

```
Podaj liczbe: 2
Liczba: dwa
Liczba: trzy
Liczba: cztery
Liczba: piec
Inna liczba
```

- Pominięcie instrukcji break spowoduje wykonanie wszystkich instrukcji występujących po danym case (do końca switch)

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:

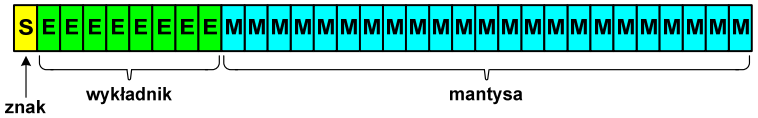
$2{,}43 \cdot 10^{3}$ - to jest postać znormalizowana, gdyż: $10 > |2{,}43| \geq 1$

$0{,}243 \cdot 10^{4}$ - 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:

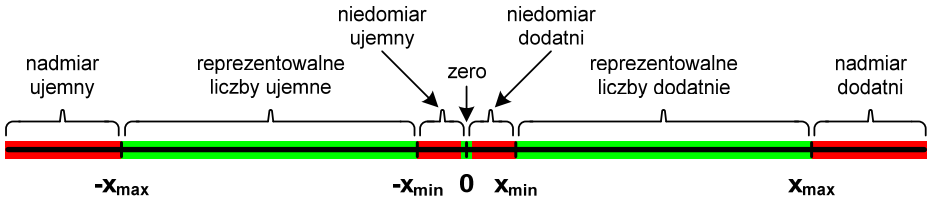
L - wartość liczby S - znak liczby M - mantysa
E - wykładnik BIAS - przesunięcie (nadmiar)

- 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:

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

Standard IEEE 754

- IEEE Std. 754-2008 - IEEE Standard for Floating-Point Arithmetic
- Standard definiuje następujące klasy liczb zmiennoprzecinkowych:

Precyzja	Długość słowa [bity]	Znak [bity]	Wykładnik		Mantysa	
			Długość [bity]	Zakres	Długość [bity]	Cyfry znaczące
Pojedyncza (Single Precision, binary32)	32	1	8	$2^{\pm 127} \approx 10^{\pm 38}$	23	7
Pojedyncza rozszerzona (Single Extended)	≥ 43	1	≥ 11	$\geq 2^{\pm 1023} \approx 10^{\pm 308}$	≥ 31	≥ 10
Podwójna (Double Precision, binary64)	64	1	11	$2^{\pm 1023} \approx 10^{\pm 308}$	52	16
Podwójna rozszerzona (Double Extended)	≥ 79	1	≥ 15	$\geq 2^{\pm 16383} \approx 10^{\pm 4932}$	≥ 63	≥ 19

źródło: Gryś S.: „Arytmetyka komputerów w praktyce". PWN, Warszawa, 2007 (str. 116).

Standard IEEE 754

- W przypadku liczb:
  - pojedynczej rozszerzonej precyzji (ang. Single Precision)
  - podwójnej rozszerzonej precyzji (ang. Double Precision)

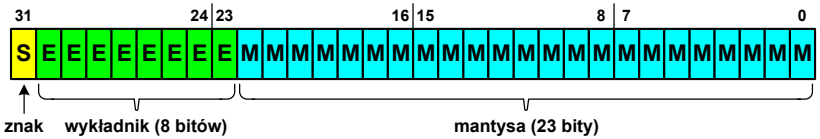
  standard podaje jedynie minimalną liczbę bitów pozostawiając szczegóły implementacji producentom procesorów i kompilatorów
- Bardzo popularny był 80-bitowy format podwójnej rozszerzonej precyzji (Extended Precision) wprowadzony przez firmę Intel
- W 80-bitowym formacie Intela:
  - długość słowa: 80 bitów
  - znak: 1 bit
  - wykładnik: 15 bitów (zakres: $2^{\pm 16383} \approx 10^{\pm 4932}$)
  - mantysa: 63 bity (cyfry znaczące: 19)

Standard IEEE 754

- Standard IEEE 754 definiuje dziesiętne typy zmiennoprzecinkowe (operujące na cyfrach dziesiętnych):
  - decimal32 (32 bity, 7 cyfr dziesiętnych)
  - decimal64 (64 bity, 16 cyfr dziesiętnych)
  - decimal128 (128 bitów, 34 cyfry dziesiętnych)
- Standard IEEE 754 definiuje:
  - sposób reprezentacji specjalnych wartości, np. nieskończoności, zera
  - sposób wykonywania działań na liczbach zmiennoprzecinkowych
  - sposób zaokrąglania liczb

Standard IEEE 754 - liczby 32-bitowe

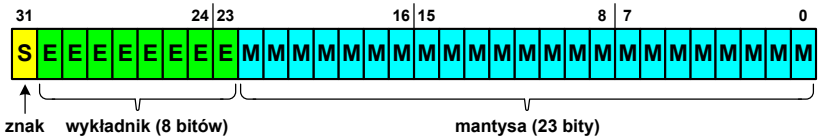
- Liczba pojedynczej precyzji przechowywana jest na 32 bitach:



- Pierwszy bit w zapisie (bit nr 31) jest bitem znaku (0 - liczba dodatnia, 1 - liczba ujemna)
- Wykładnik zapisywany jest na 8 bitach (bity nr 30-23) z nadmiarem o wartości 127
- Wykładnik może przyjmować wartości od -127 (wszystkie bity wyzerowane) do 128 (wszystkie bity ustawione na 1)

Standard IEEE 754 - liczby 32-bitowe

- Liczba pojedynczej precyzji przechowywana jest na 32 bitach:



- Mantysa w większości przypadków jest znormalizowana
- Wartość mantysy zawiera się pomiędzy 1 a 2, a zatem w zapisie liczby pierwszy bit jest zawsze równy 1
- Powyższy bit nie jest zapamiętywany, natomiast jest automatycznie uwzględniany podczas wykonywania obliczeń
- Dzięki pominięciu tego bitu zyskujemy dodatkowy bit mantysy (zamiast 23 bitów mamy 24 bity)

Standard IEEE 754 - liczby 32-bitowe

- Przykład:
  - obliczmy wartość dziesiętną liczby zmiennoprzecinkowej

$$01000010110010000000000000000000_{(\text{IEEE}754)} = ?_{(10)}$$

  - dzielimy liczbę na części

$$\underbrace{0}_{\text{S}-\text{bit znaku}} \quad \underbrace{10000101}_{\text{E}-\text{wykladnik}} \quad \underbrace{10010000000000000000000}_{\text{M}-\text{mantysa (tylko czesc ulamkowa)}}$$

  - określamy znak liczby

$$\text{S} = 0 \quad -\text{liczba dodatnia}$$

  - obliczamy wykładnik (nadmiar: 127)

$$10000101_{(2)} = 128+4+1 = 133 \quad \Rightarrow \quad \text{E} = 133 - \underbrace{127}_{nadmiar} = 6_{(10)}$$

Standard IEEE 754 - liczby 32-bitowe

- Przykład (cd.):
  - wyznaczamy mantysę dopisując na początku 1, (część całkowita)

$$\text{M} = 1{,}10010000000000000000000 =$$
$$= 1\cdot 2^0 + 1\cdot 2^{-1} + 1\cdot 2^{-4} = 1 + 0{,}5 + 0{,}0625 = 1{,}5625_{(10)}$$

  - wzór na wartość dziesiętną liczby zmiennoprzecinkowej:

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

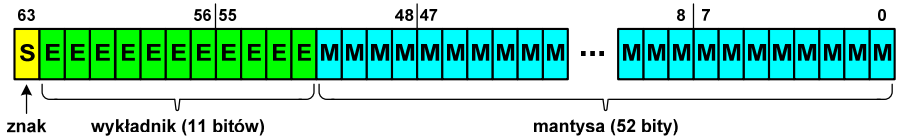
  - podstawiając otrzymujemy:

$$\text{S} = 0, \qquad \text{E} = 6_{(10)}, \qquad \text{M} = 1{,}5625_{(10)}$$
$$\text{L} = (-1)^0 \cdot 1{,}5625 \cdot 2^6 = 100_{(10)}$$

$$01000010110010000000000000000000_{(\text{IEEE}754)} = 100_{(10)}$$

Standard IEEE 754 - liczby 64-bitowe

- Liczba podwójnej precyzji przechowywana jest na 64 bitach:



- Pierwszy bit w zapisie (bit nr 63) jest bitem znaku (0 - liczba dodatnia, 1 - liczba ujemna)
- Wykładnik zapisywany jest na 11 bitach (bity nr 62-52) z nadmiarem o wartości 1023
- Wykładnik może przyjmować wartości od -1023 (wszystkie bity wyzerowane) do 1024 (wszystkie bity ustawione na 1)
- Mantysa zapisywana jest na 52 bitach (pierwszy bit mantysy, zawsze równy 1, nie jest zapamiętywany)

Standard IEEE 754 - zakres liczb

- Pojedyncza precyzja:
  - największa wartość: $\approx 3{,}4 \cdot 10^{38}$
  - najmniejsza wartość: $\approx 1{,}4 \cdot 10^{-45}$
  - zakres liczb: $<-3{,}4\cdot10^{38} \ldots -1{,}4\cdot10^{-45}> \cup \{0\} \cup <1{,}4\cdot10^{-45} \ldots 3{,}4\cdot10^{38}>$
- Podwójna precyzja:
  - największa wartość: $\approx 1{,}8 \cdot 10^{308}$
  - najmniejsza wartość: $\approx 4{,}9 \cdot 10^{-324}$
  - zakres liczb: $<-1{,}8\cdot10^{308} \ldots -4{,}9\cdot10^{-324}> \cup \{0\} \cup <4{,}9\cdot10^{-324} \ldots 1{,}8\cdot10^{308}>$
- Podwójna rozszerzona precyzja:
  - największa wartość: $\approx 1{,}2 \cdot 10^{4932}$
  - najmniejsza wartość: $\approx 3{,}6 \cdot 10^{-4951}$
  - zakres liczb: $<-1{,}2\cdot10^{4932} \ldots -3{,}6\cdot10^{-4951}> \cup \{0\} \cup <3{,}6\cdot10^{-4951} \ldots 1{,}2\cdot10^{4932}>$

Standard IEEE 754 - precyzja liczb

- Precyzja - liczba zapamiętywanych cyfr znaczących w systemie (10)

  4,86452137846 → 4,864521 - 7 cyfr znaczących
- Precyzja liczby zależy od liczby bitów mantysy
- Liczba bitów potrzebnych do zakodowania 1 cyfry dziesiętnej:

$$10^1 = 2^n \quad \rightarrow \quad n = \log_2(10) \approx 3{,}321928$$

- Liczba cyfr dziesiętnych (d) możliwa do zakodowania na m bitach:

  $\log_2(10)$ bitów - 1 cyfra dziesiętna
  m bitów - d cyfr dziesiętnych

$$d = \frac{m}{\log_2(10)}$$

Standard IEEE 754 - precyzja liczb

- Dla formatu pojedynczej precyzji:
  - mantysa: 23 + 1 = 24 bity
  - cyfry znaczące: 7

$$d = \frac{24}{\log_2(10)} = \frac{24}{3{,}321928} = 7{,}2247 \approx 7$$

- Dla formatu podwójnej precyzji:
  - mantysa: 52 + 1 = 53 bity
  - cyfry znaczące: 16

$$d = \frac{53}{\log_2(10)} = \frac{53}{3{,}321928} = 15{,}9546 \approx 16$$

- Dla formatu podwójnej rozszerzonej precyzji:
  - mantysa: 63 + 1 = 64 bity
  - cyfry znaczące: 19

$$d = \frac{64}{\log_2(10)} = \frac{64}{3{,}321928} = 19{,}2659 \approx 19$$

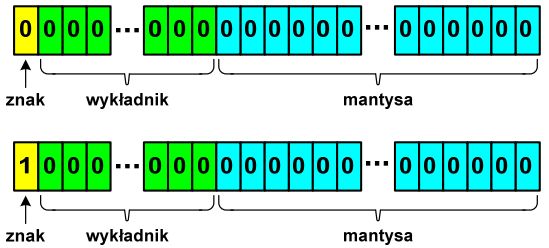
Standard IEEE 754 - precyzja liczb

```
#include <stdio.h>

int main()
{
    float x;
    double y;

    x = 1234567890.0;    /* 1.234.567.890 */
    y = 1234567890.0;    /* 1.234.567.890 */
    printf("float  -> %f\n",x);
    printf("double -> %f\n\n",y);

    y = 12345678901234567890.0;
    printf("double -> %f\n",y);

    return 0;
}
```

```
float  -> 1234567936.000000
double -> 1234567890.000000

double -> 12345678901234567000.000000
```


Standard IEEE 754 - wartości specjalne

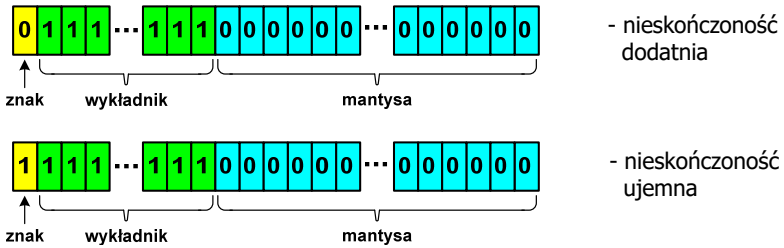
- Zero:



- Podczas porównań zero dodatnie i ujemne są traktowane jako równe sobie

Standard IEEE 754 - wartości specjalne

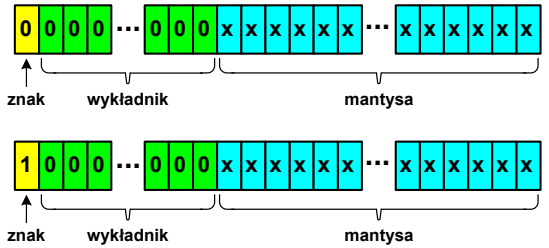
- Nieskończoność:



- Nieskończoność występuje w przypadku wystąpienia nadmiaru (przepełnienia) oraz przy dzieleniu przez zero

Standard IEEE 754 - wartości specjalne

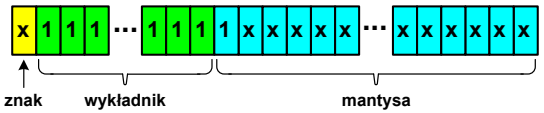
- Liczba zdenormalizowana:



- Pojawia się, gdy występuje niedomiar (ang. underflow), ale wynik operacji można jeszcze zapisać denormalizując mantysę
- Mantysa nie posiada domyślnej części całkowitej równej 1, tzn. reprezentuje liczbę o postaci 0,xxx…xxx, a nie 1,xxx…xxx

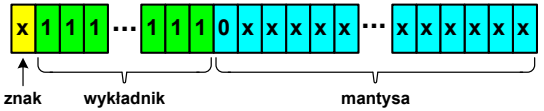
Standard IEEE 754 - wartości specjalne

- Nieliczby - NaN (Not A Number) - nie reprezentują wartości liczbowej
- Powstają w wyniku wykonania niedozwolonej operacji
- QNaN (ang. Quiet NaN) - ciche nieliczby



  - „przechodzą” przez działania arytmetyczne (brak przerwania wykonywania programu)

- SNaN (ang. Signaling NaN) - sygnalizujące, istotne, głośne nieliczby



  - zgłoszenie wyjątku (przerwanie wykonywania programu)

Koniec wykładu nr 4

Dziękuję za uwagę!