Reprezentacja symboli w komputerze.

Liczby całkowite i zmiennoprzecinkowe.

Ostatnia aktualizacja: 12 stycznia 2026

Bity i kody binarne

- **Bit** (***bi****nary* ***d****igit*) najmniejsza ilość informacji $\{0, 1\}$, wysokie/niskie napięcie
- **Kod binarny**- grupa bitów reprezentująca zbiór symboli

  1b $\rightarrow$ 2 symbole $\{0, 1\}$

  2b $\rightarrow$ 4 symbole $\{00, 01, 10, 11\}$

  8b $\rightarrow$ 256 = 2^8 symboli

  n bitów $\rightarrow$ 2^n symboli
- Do zakodowania n symboli potrzeba co najmniej $\lceil \log_2 n \rceil$ bitów
- Ile bitów potrzeba do zakodowania alfabetu polskiego?

Bajt (*byte*) 1B = 8 b

- wg. standardu C: najmniejsza porcja adresowalnej pamięci
- stała `CHAR_BIT` z pliku `limits.h`
- *Najmniejsza porcja danych, którą może „ugryźć” komputer.*
- utożsamiany ze znakiem (zmienna typu `char` ma rozmiar 1B).

Ile pamięci można zaadresować zależnie od szerokości słowa bitowego adresu?

16b $\rightarrow 2^{16}$B = 64 KB
32b $\rightarrow 2^{32}$B = 4294967296B $\sim$ 4GB
64b $\rightarrow 2^{64}$B $\approx$ 16EB (eksa bajty)

Przedrostki jednostek

- układ SI: $1k = 10^3 = 1000 \neq 1024 = 2^{10} = 1K$
- przedrostki dwójkowe zgodnie z normą Międzynarodowej Komisji Elektrotechnicznej (IEC, 1998)

IEC		podstawa						SI	
nazwa	symbol	2	16		różnica	10		nazwa	symbol
kibi	Ki	2^{10}	$16^{2.5}$	400_{16}	2,40%	1 024	$> 10^3$	kilo	k
mebi	Mi	2^{20}	16^{5}	$10\ 0000_{16}$	4,86%	1 048 576	$> 10^6$	mega	M
gibi	Gi	2^{30}	$16^{7.5}$	$4000\ 0000_{16}$	7,37%	1 073 741 824	$> 10^9$	giga	G
tebi	Ti	2^{40}	16^{10}	$100\ 0000\ 0000_{16}$	9,95%	1 099 511 627 776	$> 10^{12}$	tera	T
pebi	Pi	2^{50}	$16^{12.5}$	$4\ 0000\ 0000\ 0000_{16}$	12,59%	1 125 899 906 842 624	$> 10^{15}$	peta	P
eksbi	Ei	2^{60}	16^{15}	$1000\ 0000\ 0000\ 0000_{16}$	15,29%	1 152 921 504 606 846 976	$> 10^{18}$	eksa	E
zebi	Zi	2^{70}	$16^{17.5}$	$40\ 0000\ 0000\ 0000\ 0000_{16}$	18,06%	1 180 591 620 717 411 303 424	$> 10^{21}$	zetta	Z
jobi	Yi	2^{80}	16^{20}	$1\ 0000\ 0000\ 0000\ 0000\ 0000_{16}$	20,89%	1 208 925 819 614 629 174 706 176	$> 10^{24}$	jotta	Y

https://pl.wikipedia.org/wiki/Przedrostek_dwójkowy

Operator sizeof

Operator `sizeof` zwraca rozmiar typu lub zmiennej w bajtach

Składnia

`sizeof (`*typ*`)`
`sizeof` *zmienna*

Przykład

```
int x = 10;
printf("%d\n", sizeof (int));
printf("%d\n", sizeof x);
```

Typowe użycie

```
int *x;
x = malloc(sizeof(int) * n);
```

```c
#include <stdio.h>
#include <limits.h>

struct s f
    int a;
    char b[100];
g;

int main()
f
    int tab[5];

    printf("CHAR_BIT          = %d\n", CHAR_BIT);
    printf("sizeof (char)     = %d\n", sizeof (char));
    printf("sizeof (int)      = %d\n", sizeof (int));
    printf("sizeof (long)     = %d\n", sizeof (long));
    printf("sizeof (float)    = %d\n", sizeof (float));
    printf("sizeof (double)   = %d\n", sizeof (double));
    printf("sizeof (int *)    = %d\n", sizeof (int*));
    printf("sizeof (char *)   = %d\n", sizeof (char*));
    printf("sizeof (struct s) = %d\n", sizeof (struct s));
    printf("sizeof tab        = %d\n", sizeof tab);

    return 0;
g
```

sizeof.c

```c
#include <stdio.h>
#include <limits.h>

struct s f
    int a;
    char b[100];
g;

int main()
f
    int tab[5];

    printf("CHAR_BIT          = %d\n", CHA
    printf("sizeof (char)     = %d\n", siz
    printf("sizeof (int)      = %d\n", siz
    printf("sizeof (long)     = %d\n", sizeof (long));
    printf("sizeof (float)    = %d\n", sizeof (float));
    printf("sizeof (double)   = %d\n", sizeof (double));
    printf("sizeof (int *)    = %d\n", sizeof (int*));
    printf("sizeof (char *)   = %d\n", sizeof (char*));
    printf("sizeof (struct s) = %d\n", sizeof (struct s));
    printf("sizeof tab        = %d\n", sizeof tab);

    return 0;
g
```

Przykładowy wynik (może się różnić zależnie od architektury komputera)

```
CHAR_BIT          = 8
sizeof (char)     = 1
sizeof (int)      = 4
sizeof (long)     = 8
sizeof (float)    = 4
sizeof (double)   = 8
sizeof (int *)    = 8
sizeof (char *)   = 8
sizeof (struct s) = 104
sizeof tab        = 20
```

sizeof.c