

Programmētāju skola

3. līmeņa grupa

Skaitļošanās sistēmas

Skaitļošanas sistēmas

Skaitļošanas sistēma – tas ir skaitļu pieraksta veids ar simbolu (ciparu) palīdzību. Tradicionāli matemātikā pielieto desmit ciparus. Tādu sistēmu dēvē par decimālo. Bet var izbūvēt sistēmas ar lielāku vai mazāku ciparu skaitu.

10	5	2	8	16
0	0	0	0	0
1	1	1	1	1
2	2	10	2	2
3	3	11	3	3
4	4	100	4	4
5	10	101	5	5
6	11	110	6	6
7	12	111	7	7
8	13	1000	10	8
9	14	1001	11	9
10	20	1010	12	A
11	21	1011	13	B
12	22	1100	14	C
13	23	1101	15	D
14	24	1110	16	E
15	30	1111	17	F
16	31	10000	20	10
17	32	10001	21	11
18	33	10010	22	12
19	34	10011	23	13
$20_{10} = 40_5 = 10100_2 = 24_8 = 14_{16}$				

Svērtās sistēmas

Katra cipara nozīmīgums (svars) tiek noteikts ar tās pozīciju skaitlī: jo tālāk skaitlis ir ierakstīts pa kreisi, jo lielāks ir tā svars. Tādas sistēmas dēvē par svērtām sistēmām.

$$\begin{array}{ccccccc} & n & & 3 & 2 & 1 & 0 \\ N & . & . & . & 1 & 5 & 4 & 3 &_{10} & = \\ 10^n & & & & 1000 & 100 & 10 & 1 \end{array}$$

$$= 3 \cdot 1 + 4 \cdot 10 + 5 \cdot 100 + 1 \cdot 10^3 + \dots + N \cdot 10^n$$

Pārveidošana uz decimālo sistēmu

$$\begin{array}{cccccc} 1 & 0 & 1 & 0 & 0 & \\ 16 & 8 & 4 & 2 & 1 & \end{array} 2 = 1 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 0 \cdot 1 = 20_{10}$$

$$\begin{array}{ccc} 1 & 3 & 4 \\ 25 & 5 & 1 \end{array} 5 = 1 \cdot 25 + 3 \cdot 5 + 4 \cdot 1 = 44_{10}$$

$$\begin{array}{ccc} 2 & 0 & 7 \\ 64 & 8 & 1 \end{array} 8 = 2 \cdot 64 + 0 \cdot 8 + 7 \cdot 1 = 135_{10}$$

$$\begin{array}{ccc} A & 2 & 0 \\ 256 & 16 & 1 \end{array} 16 = 10 \cdot 256 + 2 \cdot 16 + 0 \cdot 1 = 2592_{10}$$

$$10 \ 0011_2 = 1 + 2 + 32 = 35_{10}$$

$$1001 \ 0101_2 = 1 + 4 + 16 + 128 = 149_{10}$$

$$1 \ 0000 \ 0000_2 = 2^8 = 256_{10}$$

$$1111 \ 1111_2 = 2^8 - 1 = 255_{10}$$

$$1111 \ 0111_2 = 255 - 8 = 247_{10}$$

Informācijas mērvienības

0 vai 1	bits
8 baiti	baiti
2 baiti	vārds
$2^{10}=1024$ baiti	kilobaiti, KB
2^{20} baiti	megabaiti, MB
2^{30} baiti	gigabaiti, GB
2^{40} baiti	terabaiti, TB
2^{50} baiti	petabaiti, PB
2^{60} baiti	eksabaiti, EB

$$2^{24} \text{ baiti} = 2^4 \cdot 2^{20} \text{ baiti} = 16\text{MB}$$

$$2^{45} \text{ baiti} = 2^5 \cdot 2^{40} \text{ baiti} = 32\text{TB}$$

$$2^{62} \text{ baiti} = 2^2 \cdot 2^{60} \text{ baiti} = 4\text{EB}$$

Skaitļa sadalīšana ciparos

Dažos gadījumos (piemēram, izvadei uz ekrāna) mums ir jāzina, kādi cipari veido skaitli.

$$12345_{10} = 5, 4, 3, 2, 1$$

$$12345 / 10 = 1234 : 5$$

$$1234 / 10 = 123 : 4$$

$$123 / 10 = 12 : 3$$

$$12 / 10 = 1 : 2$$

$$1 / 10 = 0 : 1$$

Pārveidošana no decimālās sistēmas

$$20_{10} = 10100_2$$

$$\begin{array}{r|l} 20/2=10 & 0 \\ 10/2=5 & 0 \\ 5/2=2 & 1 \\ 2/2=1 & 0 \\ 1/2=0 & 1 \end{array} \uparrow$$

$$120_{10} = 440_5$$

$$\begin{array}{r|l} 120/5=24 & 0 \\ & 4 \\ & 0 \end{array} \uparrow$$

$$295_{10} = 447_8$$

$$\begin{array}{r|l} 295/8=36 & 7 \\ & 4 \\ & 0 \end{array} \uparrow$$

$$395_{10} = 18B_{16}$$

$$\begin{array}{r|l} 395/16=24 & 11 \\ & 1 \\ & 0 \end{array} \uparrow$$

Pārveidošana starp bināro un heksadecimālo sistēmām

$$\underbrace{1001}_9 \underbrace{1010}_{10} \underbrace{1110}_{14} \underbrace{0011}_3 \underbrace{0101}_5_2 = 9AE35_{16}$$

$$7B0F4_{16} = \underbrace{0111}_7 \underbrace{1011}_{11} \underbrace{0000}_0 \underbrace{1111}_{15} \underbrace{0100}_4_2$$

Skaitļošanās sistēmu atbalsts C valodā

Veselo skaitļu konstantes

Decimālā forma	123
Heksadecimālā forma	0x7B
Oktālā forma	0173
Binārā forma	0b01111011

cout izvads

```
#include <iostream>
#include <iomanip>
#include <bitset>

int a = 31503;
cout << a; // 31503
cout << hex << a; // 7b0f
cout << oct << a; // 75417
cout << dec << a; // 31503
cout << bitset<16>{unsigned (a)}; // 0111101100001111
```

Skaitļošanās sistēmu atbalsts C valodā

stdio izvads

```
int a = 31503;
printf("%x", a);    // 7b0f
printf("%X", a);    // 7B0F
printf("%o", a);    // 75417
printf("%d", a);    // 31503
printf("%i", a);    // 31503
```

```
printf(
    "%s",
    std::bitset<16>{unsigned(a)}.to_string().c_str()
);
// 0111101100001111
```

Loģiskās operācijas

Loģiskā saskaitīšana
operācija **VAI**

a	b	a or b
0	0	0
0	1	1
1	0	1
1	1	1

Loģiskā reizināšana
operācija **UN**

a	b	a and b
0	0	0
0	1	0
1	0	0
1	1	1

Loģiskais noliegums
operācija **NE**

a	not a
0	1
1	0

Ekskluzīva **VAI** operācija

$a \text{ xor } b = \text{not } a \text{ and } b \text{ or } a \text{ and not } b$

a	b	a xor b
0	0	0
0	1	1
1	0	1
1	1	0

Loģisko operāciju īpašības

not not $a = a$
 $a \text{ or } 1 = 1$
 $a \text{ or } 0 = a$
 $a \text{ or } a = a$
 $a \text{ or not } a = 1$

$a \text{ and } 1 = a$
 $a \text{ and } 0 = 0$
 $a \text{ and } a = a$
 $a \text{ and not } a = 0$
 $a \text{ xor } 1 = \text{not } a$
 $a \text{ xor } 0 = a$

Loģiskās bitu operācijas C valodā

Lai aprēķinātu loģiskas bitu operācijas rezultātu, nepieciešams operandus uzrakstīt binārā veidā vienu virs otra un izpildīt loģisko operāciju ar katru bitu pāri.

$$27 \text{ and } 50 = 18$$

$$27 = 0001 \ 1011$$

$$50 = 0011 \ 0010$$

$$\underline{\hspace{1cm}} \\ 0001 \ 0010 = 18$$

Bitu operācija **AND** $27 \ \& \ 50 = 18$

Bitu operācija **OR** $27 \ |\ 50 = 59$

Bitu operācija **XOR** $27 \ \wedge \ 50 = 41$

Bitu operācija **NOT** (inversija) $\sim 50 = 205$

Nobīde pa kreisi **SHL** $50 \ \ll \ 2 = 200$

Nobīde pa labi **SHR** $50 \ \gg \ 2 = 12$

$$\begin{array}{r} 0001 \ 1011 \\ 0011 \ 0010 \\ \hline \& = 0001 \ 0010 \\ | = 0011 \ 1011 \\ \wedge = 0010 \ 1001 \end{array}$$

$$\begin{array}{r} 0011 \ 0010 \\ \hline \sim = 1100 \ 1101 \end{array}$$

$$\begin{array}{r} \leftarrow \\ 0011 \ 0010 \\ \hline \ll 2 = 1100 \ 1000 \end{array}$$

$$\begin{array}{r} \rightarrow \\ 0011 \ 0010 \\ \hline \gg 2 = 0000 \ 1100 \end{array}$$

Operācijas ar bitiem

Pielietojot loģiskās bitu operācijas, var apstrādāt atsevišķus skaitļa bitus, nepārveidojot skaitli uz bināro formu.

Iestatīt bitu ar numuru N skaitlī A

$$\begin{aligned} A &= \text{aaaa aaaa} \\ \text{mask} &= \underline{0001\ 0000} = 1 \ll N & A \mid &= 1 \ll N \\ A \mid \text{mask} &= \text{aaa}\mathbf{1}\text{ aaaa} = A \mid 1 \ll N \end{aligned}$$

Attīrīt bitu ar numuru N skaitlī A

$$\begin{aligned} A &= \text{aaaa aaaa} \\ \text{mask} &= \underline{1110\ 1111} = \sim (1 \ll N) & A \&= & \sim (1 \ll N) \\ A \& \text{mask} &= \text{aaa}\mathbf{0}\text{ aaaa} = A \& \sim (1 \ll N) \end{aligned}$$

Pārbaudīt bitu ar numuru N skaitlī A

$$\begin{aligned} A &= \text{aaaa aaaa} \\ \text{mask} &= \underline{0001\ 0000} = 1 \ll N & \text{if } (A \& 1 \ll N \neq 0) \dots \\ A \& \text{mask} &= \underline{000}\mathbf{a}\underline{0000} = A \& 1 \ll N \end{aligned}$$

Invertēt (izmainīt uz pretēju) bitu ar numuru N skaitlī A

$$\begin{aligned} A &= \text{aaaa aaaa} \\ \text{mask} &= \underline{0001\ 0000} = 1 \ll N & A \wedge &= 1 \ll N \\ A \wedge \text{mask} &= \text{aaa}\mathbf{\bar{a}}\text{ aaaa} = A \wedge 1 \ll N \end{aligned}$$

Karodziņu kopas

```
const int DO_1 = 1 << 0;
const int DO_2 = 1 << 1;
const int DO_3 = 1 << 2;

void func(int doFlg) {
    if (doFlg & DO_1) do_something_1;
    if (doFlg & DO_2) do_something_2;
    if (doFlg & DO_3) do_something_3;
}

int main() {
    func(DO_1);
    func(DO_1 | DO_2);
    func(DO_1 | DO_2 | DO_3)
}
```

Piemērs

Izveidot masīvu uz ekrāna, izceļot nulles, minimālo un maksimālo vērtību pēc lietotāja izvēles.

```
const int HL_NONE = 0;
const int HL_MAX = 1 << 0;
const int HL_MIN = 1 << 1;
const int HL_ZEROS = 1 << 2;

void showArray(int array[], int count, int highlight) {
    int max = highlight & HL_MAX ? getMax(array, count) : INT_MAX;
    int min = highlight & HL_MIN ? getMin(array, count) : INT_MAX;
    int zero = highlight & HL_ZEROS ? 0 : INT_MAX;

    for (int i = 0; i < count; i++) {
        if (array[i] == min) cout << "\x1b[91m";
        else if (array[i] == max) cout << "\x1b[92m";
        else if (array[i] == zero) cout << "\x1b[94m";
        else cout << "\x1b[0m";
        cout << array[i] << "\t";
    }
    cout << "\n\x1b[0m";
}
```

Piemērs

Izveidot masīvu uz ekrāna, izceļot nulles, minimālo un maksimālo vērtību pēc lietotāja izvēles.

```
#include <iostream>

using namespace std;

int getMin(int array[], int count) { ... }
int getMax(int array[], int count) { ... }

const int HL_NONE    = 0;
const int HL_MAX     = 1 << 0;
const int HL_MIN     = 1 << 1;
const int HL_ZEROS   = 1 << 2;
void showArray(int array[], int count, int highlight) { ... }

int main(){
    int a[] = { 12, 5, -3, 54, 0, 59, -34, -76, 0, 23 };
    showArray(a, _countof(a), HL_NONE);
    showArray(a, _countof(a), HL_ZEROS);
    showArray(a, _countof(a), HL_MIN | HL_MAX);
}
```