Лекція № 2 Логічні та побітові операції

Лектор Нечипорук О.П.

Операції відношення, логічні операції і логічні вирази

- Строго кажучи, логічне значення "істина" відповідає будь-якому числовому значенню, відмінному від нуля. Саме така домовленість прийнята в мовах C, PHP, Perl, Java
- Це дає можливість об'єднати поняття арифметичного, умовного і логічного виразів у єдиному понятті "вираз", що дуже важливо з точки зору гнучкості і "симетричності" мови.

Логічні операції і логічні вирази

- При розробці реальних програм часто виявляється необхідним об'єднати два або більш умовних виразів.
- Це можна зробити, використовуючи набір двохмістних логічних операцій:
- && - логічне І
- || - логічне АБО
- ! - логічне НІ (заперечення)

Логічні операції і логічні вирази

- Припустимо, що expression1 і expression2 - два простих умовних вирази. Тоді:
- 1. значення expression1 && expression2 є істиною тоді і тільки тоді, коли обидва вирази expression1 і expression2 істинні;
- 2. значення expression1 || expression2 є істиною, якщо хоча б один з виразів-операндів має значення "істина";
- 3. значення !expression1 є істиною, якщо вираз expression1 є не істиною, і навпаки.

Логічні операції і логічні вирази

- Вирази, побудовані з використанням логічних операцій, ми будемо називати логічними виразами.
- Логічні вирази є прямим узагальненням простих умовних виразів.
- Стандартний порядок їхньої обробки – з ліва на право. Пріоритет логічних операцій && і || нижче пріоритету будь-якої операції відношення і тому логічні вирази

  a < b && b < c і (a < b) && (b < c)
- цілком рівносильні, хоча друге з них є більш кращим через наочність.

Логічні операції і логічні вирази

ОДНАК

операція логічного заперечення (!) має дуже високий пріоритет (він такий же, як пріоритет одномісних арифметичних операцій) і тільки круглі дужки мають більш високий пріоритет.

Логічні операції і логічні вирази

- У загальному випадку операндами логічних операцій можуть бути не тільки умовні вирази, але і будь-які арифметичні вирази. Це легко зрозуміти, якщо нульовому значенню арифметичного виразу поставити у відповідність логічне значення "не істина" і, навпаки, всяке відмінне від нуля числове значення ототожнити з логічним значенням "істина".

Умовний оператор

- Найпростішою інструкцією мови Сі, що використовує логічні вирази, є умовний оператор:

expression1 ? expression2 : expression3

- де expression1 це логічне вираз, а expression2 і expression3 це довільні арифметичні вирази.

Умовний оператор

expression1 ? expression2 : expression3

- Якщо expression1 приймає значення "істина", то результатом умовної операції буде значення expression2, у противному випадку він дорівнює значенню expression3.

- Наприклад, інструкція
- abs_a = (a > 0) ? a : -a
- привласнює змінній abs_a абсолютне значення змінної a

Побітові операції

- Побітові операції – це операції,які передбачають прямі дії з бітами змінних, або визначеними бітами комірок пам’яті.
- Порозрядні операції застосовуються тільки до цілочисельних операндів і "працюють" з їх двійковими представленнями. Ці операції неможливо використовувати із змінними типу double, float, long double.

Особливість переведення чисел до двійкової системи

- Продовжіть ряд:
  - 1
  - 2
  - 4
  - 8
  - 16
  - 32
  - ...
  - 1024
  - 2^0
  - 2^1
  - 2^2
  - 2^3
  - 2^4
  - 2^5
  - ...
  - 2^{10}

Особливість переведення чисел до та з двійкової системи

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

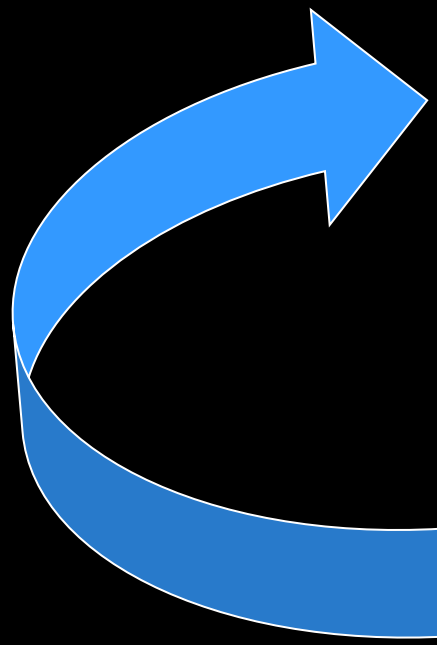
$1*2^0+0*2^1+1*2^2+0*2^3+\ldots=5$

0	1	0	1	1	1	1	1

$1*2^0+1*2^1+1*2^2+1*2^3+1*2^4+0*2^5+1*2^6+0*2^7+\ldots=95$

Двійкова система – тест на швидкість

- Знайдіть двійкове представлення:
  - 1
  - 6
  - 9
  - 11
  - 12
  - 15
  - 25



  - 0001
  - 0110
  - 1001 = 8+0+0+1
  - 1011 = 8+0+2+1
  - 1100 = 8+4+0+0
  - 1111 = 8+4+2+1
  - 1 1001 = 16+9

Побітові операції

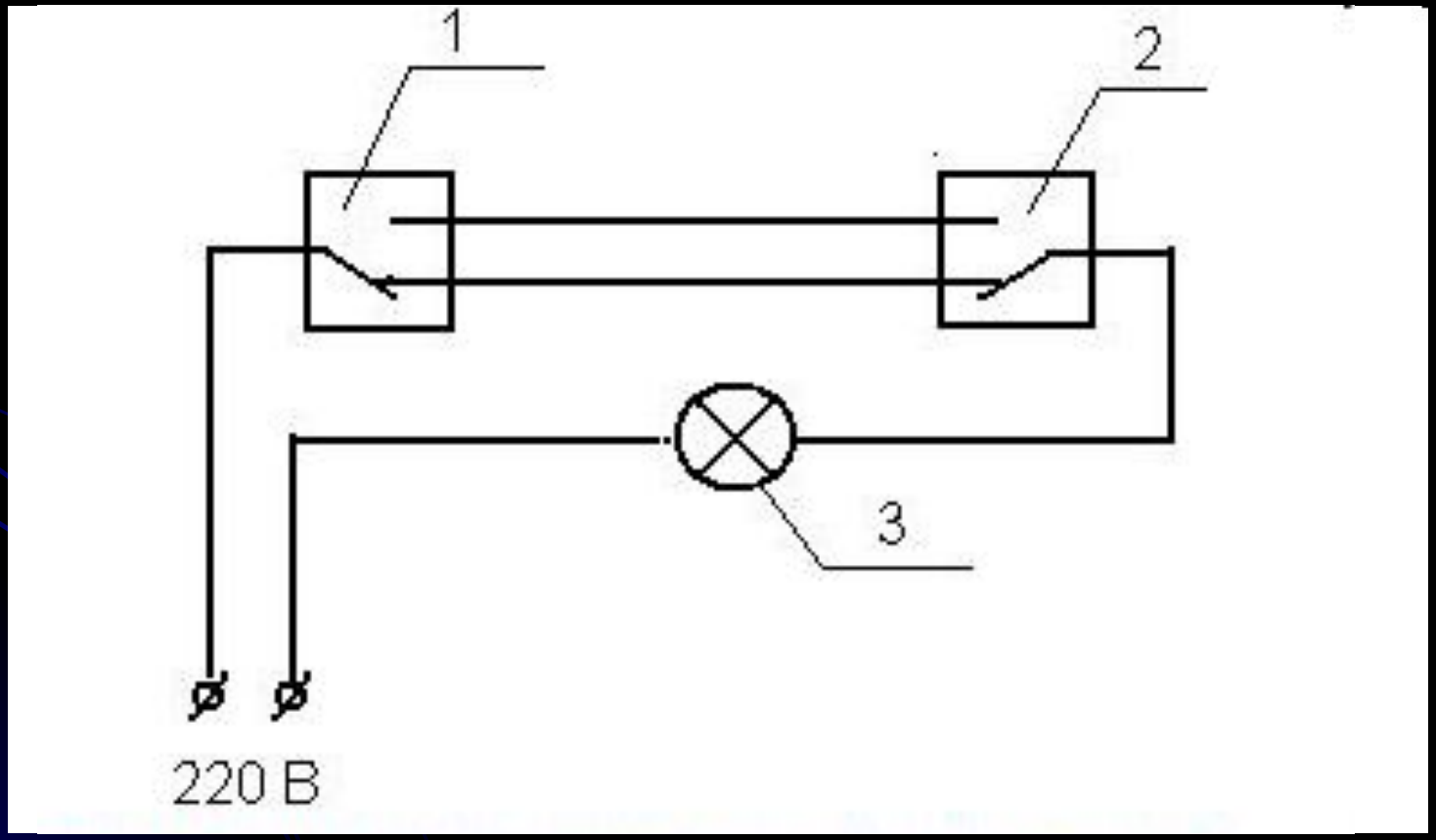
Операція	Значення
~	порозрядне заперечення
&	побітова кон'юнкція (побітове І)
|	побітова диз'юнкція (побітове АБО)
^	побітове додавання за МОД2
<<	зсув вліво
>>	зсув вправо

Таблиця істинності логічних побітових операцій

E1	E2	E1&E2	E1^E2	E1|E2
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	0	1

Приклад XOR у реальному світі

Прохідний перемикач



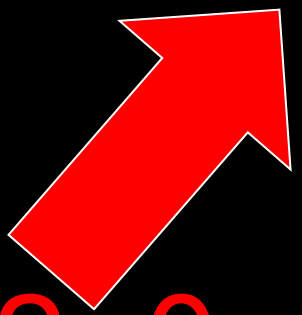
Обчислення побітових операцій

- Побітове І:

5&4=4

7&3=3

5&2=???



5&2=0

Обчислення побітових операцій

- Побітове І:

  5&4=4

  7&3=3

  13&14=12

Особливість переведення чисел до двійкової системи

- Продовжіть ряд:
  - 1
  - 3
  - 7
  - 15
  - 31
  - 63
  - ...
  - 1023
  - 2^1-1
  - 2^2-1
  - 2^3-1
  - 2^4-1
  - 2^5-1
  - 2^6-1
  - ...
  - $2^{10}-1$

Особливість переведення чисел до двійкової системи

- Трикутник з одиниць:
  - 1
  - 3
  - 7
  - 15
  - 31
  - 63
  - ...
  - 1023
  - 1
  - 11
  - 111
  - 1111
  - 11111
  - 111111
  - ...
  - 1111111111

Особливість переведення чисел до двійкової системи

- Трикутник з одиниць:
  - 0000000001 = 1
  - 0000000011 = 3
  - 0000000111 = 7
  - 0000001111 = 15
  - 0000011111 = 31
  - 0000111111 = 63
  - …
  - 1111111111 = 1023

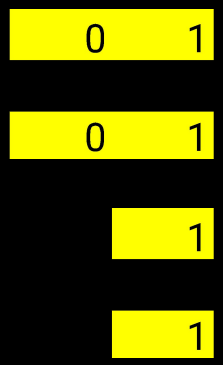
Обчислення побітових операцій

- Побітове І – виділення молодших розрядів:

5&7=5

5&3=1

5&1=1



Обчислення побітових операцій

- Побітове АБО:

  5|4=5

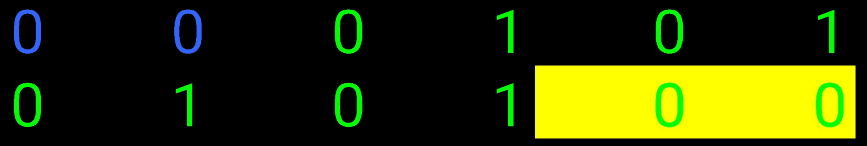
  7|3=7

  13|14=15

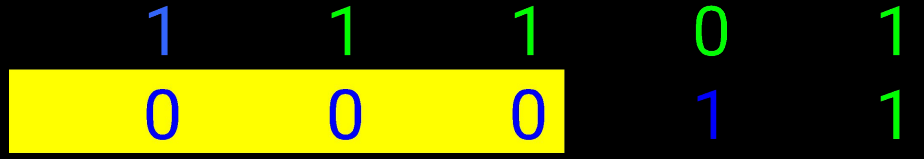
Обчислення побітових операцій

- Побітові зсуви (<< та >>):

5<<2=101 << 2 = 10100 = 20



29>>3=11101>>3=00011 = 3



Приклад стиснення даних

- Вік -> 7 біт ->128 років (age - A)
- Стать ->1 біт (0 – Ж, 1 – Ч) (sex - S)
- Сімейний стан ->2 біти (00 – неодружений(на), 01 – одружений(на), 10 – розлучений(на), 11 – вдовець/вдова) (family - F)
- Кількість дітей -> 4 біт (максимум 15) (child - C)

int K=CCCCFFSAAAAAAA;

Приклад стиснення даних

- Вік -> 7 біт ->128 років (age - A)
- Стать ->1 біт (0 – Ж, 1 – Ч) (sex - S)
- Сімейний стан ->2 біти (00 – неодружений(на), 01 – одружений(на), 10 – розлучений(на), 11 – вдовець/вдова) (family - F)
- Кількість дітей -> 4 біт (максимум 15) (child - C)

int K=CCCCFFSAAAAAAA;

Приклад стиснення даних

- K=CCCCFFSAAAAAAAA;

- Спосіб №1

```
int K=0,a,s,f,c;
cin>>a>>s>>f>>c;
K=a;
//заповнили К значенням а.
K=K|(s<<7);
//зсунули вліво і заповнили
//створену комірку значенням S.
K=K|(f<<8);
K=K|(c<<10);
cout<<K;
```

Спосіб №2

```
int K=0,a,s,f,c;
cin>>a>>s>>f>>c;
K=C;
K=(K<<2)|f;
K=(K<<1)|S
K=(K<<7)|a;
cout<<K;
```

Приклад стиснення даних (типизоване введення)

```
int c, f, b, n;
unsigned int UnitStateWord;
clrscr();
printf("Enter number \n");
scanf("%d %d %d %d",&a,&s,&f,&c);
UnitStateWord=(c&0xF)<<10;
printf("\n %x\n",UnitStateWord);
UnitStateWord|=(f&3)<<8;
printf("\n %x\n",UnitStateWord);
UnitStateWord|=(s&1)<<7;
printf("\n %x\n",UnitStateWord);
UnitStateWord|=a&0x3F;
printf("\n %04x\n",UnitStateWord);
```

Приклад вилучення даних з коду

- K=CCCCFFSAAAAAAA;

- int K=0,a,s,f,c;
- cin>>K;
- a=K&127;
- s=(K>>7)&1;
- f=(K>>8)&3;
- c=(K>>10)&15;
- cout<<K;

Використання побітових операцій для представлення чисел у двійковій системі

```
for (i=15;i>=0;i--)
cout<<(K>>i)&1;
```

Використання вбудованих функцій:

```
long N; char a[33];
ltoa(N,a,2); /* переведення до двійкової системи */
printf("\nУ двійковому представленні %ld = %s",N,a);
ltoa(N,a,8); /* восьмирична система */
printf("\nВ восьмиричному представленні %ld = %s",N,a);
ltoa(N,a,16); /* переведення до шістьнадцяткової системи */
printf("\nВ шістьнадцятковому представленні %ld = %s",N,a);
```

Зміна регістру літер побітови операціями (C#)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace _1_bit
{class Program
{public static char RevLetter(char text)
{
//Якщо зазирнути в таблицю ASCII кодів літер,
//то можна помітити,що малі літери від великих відрізняються лише на 1 біт
const char mask = (char)32; //00100000
// Згадуємо дискретку і дію XOR. 0 xor 1 = 1, 1 xor 1 = 0.
//Тобто 6 біт буде постійно змінюватись завдяки xor з маскою-літера змінюватиме регістр
return text ^= mask;
}
static void Main(string[] args)
{
Console.WriteLine(RevLetter('A'));
Console.WriteLine(RevLetter('s'));
Console.ReadKey();}
}
```

Дякую за увагу!!!
Зустрінемось на наступній лекції!!!