ШАГ компьютерная академия

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ С

Dərs №17

Bit əməliyyatları

Mündəricat

1. İkilik hesablama haqqında qısa məlumat **3**
Bir neçə say sistemlərinin istifadəsi 3
16-lıq say sistemi 8
2. Bit əməliyyatları **10**
3. Birləşmələr **17**
Misal 17
Nəticələr 19
4. Bit sahələri **20**
5. Ev tapşırığı **22**

1. İkilik hesablama haqqında qısa məlumat

Bir neçə say sistemlərinin istifadəsi

İkilik say sistemi

Biz artıq çoxdan bilirik ki, bitlər kompüter tərəfindən ancaq 0 və 1 kimi tanınır, və kompüter 2 əsaslı say sistemində və ya ikilik say sistemində çalışır.

Qeyd: *Bit ingilis sözündən əmələ gəlib* ***Binary Digit (ikilik rəqəm)*** *deməkdir.*

İkilik say sisteminin işləmə qaydaları ilə taniş olma vaxtı gəlib çatdı. Bu dərsimizdə biz bunları müzakirə edəcəyik.

İkilik say sistemi (bit) vasitəsi ilə istənilən bir qiyməti ala bilərik. İkilik say sistemində qiymət hər bitin nisbi mövqeyi ilə və bir rəqəmini təşkil edən bitlərin varlığı ilə təyin olunur. Aşağıda səkkiz bitli ədəd verilib, o bir rəqəmli bitlərdən ibarətdir:

```
Dəyər:      128   64   32   16   8   4   2 1
Bitlər:       1    1    1    1   1   1   1 1
```

Sağdan birinci rəqəmin dərəcəsi birdir, növbəti soldakı rəqəm - 2, növbəti - 3 və s. Səkkizdənəli birli bitlərin ümumi cəmi 255 ibarətdir.

```
(1+2+4+8+16+32+64+128=255)
```

Cəmləmə

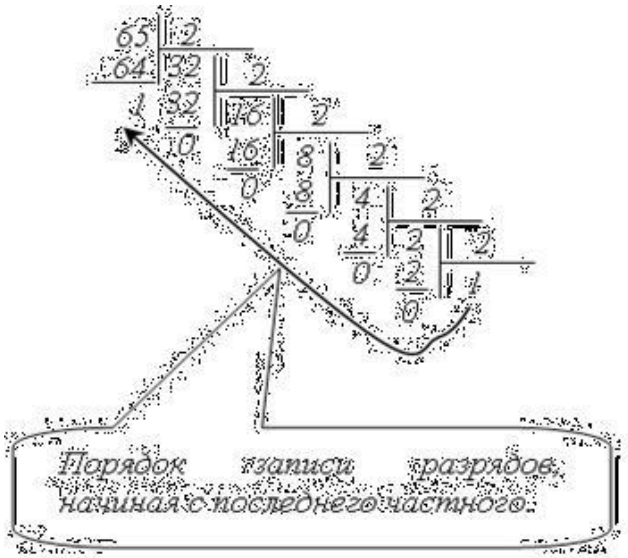
Kompüter hesab əməllərini ancaq ikilik formatında yerinə yetirir. Bu səbəbdən biz ikilik say sistemində cəmləməni bilməliyik. Gəlin onları xatırladaq:

```
0 + 0 = 0
1 + 0 = 1
1 + 1 = 10
```

Gəlin bu qaydaların istifadəsinə xüsusi misalda baxaq.

Misal: *65 və 42 ədədlərini cəmləmək, ikilik say sistemində təsvir etmək.*

Onluq say sistemində hesablama kifayət qədər asandır: 65+42=107. İkilik say sistemində bu ədədləri cəmləmək üçün şəkildə göstərildiyi kimi. ən əvvəl onları həmin say sisteminə çevirmək lazımdır:



Beləliklə: 6510 = 010000012. Diqqət yetirin ki, əvvəldə olan sıfır rəqəmi səkkiz bitdən ibarət olan ikilik ədədi tamamlamaq üçün istifadə olunur. Analoji olaraq: 4210 = 001010102. Gəlin bu ədədələri cəmləyək:

```
   01000001
+
   00101010
   --------
   01101011
```

Özünüz yenidən yoxlayib və əmin ola bilərsiniz ki, 011010112 = 10710:

```
0*27+1*26+1*25+0*24+1*23+0*22+1*21+1*20 =
64+32+8+2+1 = 107
```

Çıxma

Hal-hazırda biz kompüterdə cəmləmə əməliyyatını nəzərdən keçirtdik. Bəs çıxma əməliyyatı necə baş verir? Çıxma əməliyyatını yerinə yetirmək üçün çıx işarəsi cəm ilə əvəz olunur, ikinci ədəd isə əks işarəli ədəd kimi götürülür. Məsələn, bu çıxma əməliyyatını yerinə yetirmək lazımdır: 65 - 42. Bunu cəmləmə ilə əvəz edək: 65 + (-42). Bəs mənfi olan ikilik ədədi necə əldə edə bilərik? Bu sualı indi araşdıracayıq.

Yuxarıda verilmiş ikilik ədədlərin hamısı müsbətdir, ədədin ən solunda (yuxarı mərtəbəsində) yerləşən rəqəm sifir dəyərli kimi qeyd olunub. Mənfi olan ikilik ədədlərdə ən yuxarı mərtəbə təklik bitindən ibarətdir.

Mənfi olan ikilik ədədi hesablamaq üçün aşağıdakı alqoritmi istifadə edə bilərik:

1. Uyğun müsbət rəqəmi götürüb, onu bitlərə çevirmək (1 rəqəmlərini 0 ilə əvəz etmək və əksinə)
2. Alınan ədədin üstünə bir rəqəmini əlavə etmək.

Verilmiş alqortmin misalda istifadəsi.

Misal 1. Verilmiş ədədi ikilik say sisteminə çevirmək –65

Xatırladaq ki, $\boldsymbol{65_{10} = 01000001_2}$. Çeviririk: ***10111110***. Alınan ədədin üstünə bir rəqəmini əlavə edirik: ***10111110+1=10111111***. Hesablanmış ədədin düzgünlüyünə əmin olmalıyıq. 65 və -65 ədədlərinin cəmi 0 etməlidir:

```
    01000001
   +
    10111111
    --------
(1) 00000000
```

Səkkiz bitlərin hamısının dəyəri 0 edir. Hələlik solda olan bir rəqəmini itmiş kimi saymalıyıq. Bu qayda bizə ikilik say sistemində çıxma əməliyyatını yerinə yetirmək üçün lazımdır:

Çıxma əməliyyatı cəmləmə ilə əvəz olunur və ikinci ədədin yerinə mənfi işarəli ədəd götürülür.

Misal 2. 65 ədədindən 42 ədədini çıxmaq

İkilik say sistemində 42 ədədi bu formadadır—00101010, –42 ədədi isə ikilik say sistemində bu formada olur: 11010110

```
      65   = 01000001
+
    (-42) = 11010110
   -----     --------
      23  (1)00010111
```

Misal 3. 00000001 verilmiş ədədin üzərinə hansı qiyməti əlavə etmək lazımdır ki, 00000000 bu nəticəni alaq?

Onluq say sistemində cavab olaraq -1 alacağıq. İkilik say sistemində isə cavab bu ədəddir 11111111:

```
     00000001
   +
     11111111
     --------
 (1) 00000000
```

Sonda isə azalan sıra ilə ədədləri ikilik say sistemində təqdim edirik:

```
.     .    .    .
 3      00000011
 2      00000010
 1      00000001
 0      00000000
-1       11111111
-2       11111110
-3       11111101

  .    .      .    .
```

16-lıq say sistemi

Təsəvvür edək ki, yaddaşda bir neçə baytın içindəki məlumatlarına baxmalıyıq.

Dörd dənə ardıcıl şəkildə olan baytların məlumatlarına baxmalıyq (iki söz), hansıların ki, ikilik dəyəri var. Bu cür uzun ədədləri qısa formada təqdim etmək üçün xüsusi metod təklif edilib, bu metoda görə hər bayt yarı bölünür, və hər yarımbayt xüsusi qiymət alır. Gəlin bu dörd baytları nəzərdən keçirdək:

```
İkilik təsvir:
  0101 1001 0011 0101 1011 1001 1100 1110
Onluq təsvir:
    5    9    3    5   11    9   12   14
```

Bəzi ədədlərin təsviri üçün iki rəqəm lazım olduğu üçün hesablama sistemini genişləndirdik:

```
10=A, 11=B, 12=C, 13=D, 14=E, 15=F
```

Bələliklə, ədədləri daha qısa formada alırıq, hansılar ki, yuxarıda qeyd olunmuş baytların təsviridir:

```
59   35   B9   CE
```

Bu hesablama sistemi 0-dan F-ə qədər rəqəmlərdən ibarətdir və bu rəqəmlərin sayı 16 olduğu üçün ona 16-lıq say sistemi deyilir. Aşağıdakı cədvəldə 0-dan 15 qədər ikilik, onluq və onaltılıq say sistemlərinin qiymətləri verilmişdir.

16-lıq hesablamaların bir neçə sadə misallarına baxaq. Qeyd edək ki, 16-lıq F ədədindən sonra 16-lıq 10 ədədi gəlir, hansı ki, 16 ədədinə bərabərdir:

```
6+4=A
5+8=D
F+1=10
F+F=1E
10+10=20
FF+1=100
```

Системы исчисления		
двоичная	десятичная	шестнадцатеричная
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

2. Bit əməliyyatları

Biz ümumi nəzəriyyə ilə tanış olduq, indi isə bit əməliyyatları haqqında C baxımından danışacağıq.

C dilində bitlərlə işləyən çoxlu əməliyyatlar var. Bunlara bit əməliyyatları deyilir:

- ■■unar əməliyyatlar:
- ■■bitlərin inversiyası (~);
- ■■binar əməliyyatlar:
- ■■bit "VƏ" (&);
- ■■bit "VƏ YA" (|);
- ■■bitin istisnası "VƏ YA" (^);
- ■■sola sürüşdürmə (<<);
- ■■sağa sürüşdürmə (>>); Bu əməliyyatları daha ətraflı izah edək.

1. ***Bitlərin inversiyası*** (mətəbəyə görə mənfiləşmə, birə qədər tamamlama) bitləri çevirir, yəni hər dəyəri 1 olan bit 0 dəyərini alır və əksinə.
2. ***Bit «VƏ»*** iki operandı siniflərə görə müqayisə edir. Cavab o vaxt 1 olur ki, nə vaxt həmin mərtəbələrin hər ikisinin dəyəri birdir. Məsələn,

```
10010011 & 00111101 = 00010001
```

belə ki, sıfırıncı və dördüncü hər iki mərtəbənin dəyəri 1-dir.

3. ***Bit «VƏ YA»*** iki operandı mərtəbələrə görə müqayisə edir. Cavab o vaxt 1 olur ki, həmin mərtəbələrin istənilən birinin dəyəri birdir. Məsələn,

```
10010011 | 00111101 = 10111111
```

çünki bütün siniflərin (altıncı istisna olmaqla) istənilən birində 1 dəyəri var.

4. ***Bitin istisnası «VƏ YA»*** iki operandı mərtəbələrə görə müqayisə edir. Cavab o vaxt 1 olur ki, ikisindən biri (hər ikisi yox) həmin mərtəbələrə görə birdir. Məsələn,

```
10010011 ^ 00111101 = 10101110
```

Qeyd edək ki, operandlarda hər mərtəbənin sıfırıncısının dəyəri birdir, onların cavabı isə 0 edir.
Yuxarıda qeyd olunmuş əməliyyatlar bitlərin yerləşməsində istifadə olunur, qalan bitlər dəyişməmiş qalırlar. Bu əməliyyatlar filtrasiya və bitləri maskalamaq üçün çox əl verişlidir.

5. ***Sola sürüşdürmə*** soldakı operandın mərtəbələrini sağda qeyd olunmuş operandın qiyməti qədər sola sürüşdürür. Boşalmış mövqelər 0 ilə doldurulur, sola sürüşmüş sol operandın sinifləri isə itirlər. Məsələn,

```
10001010 << 2 = 00101000 , (hər sinif 2 mövqe qədər sürüşdü).
```

Beləliklə, ***x<<2*** ***x***-i 2 mövqe qədər sola sürüşdürür, boşalmış mövqeləri 0 ilə doldураraq (4-ə vurma buna ekvivalentdir)

İşarəsiz dəyərlər üçün

```
10001010 >> 2 = 00100010 , (hər sinif iki mövqe qədər sürüşdü).
```

Bu iki əməliyyatlar sürüşməni yerinə yetirirlər və 2 ədədinin dərəcəsinə səmərəli vurma və bölmə əməliyyatlarını aparırlar.

Misal

```
#include  <iostream>
using      namespace
std; void main (){
      int y=02,x=03,z=01,k;

      k = x|y&z;
        cout<<k<<" ";  /* əməliyyat  1 */
      k = x|y&~z;
        cout<<k<<" ";  /* əməliyyat  2 */
      k = x^y&~z;
        cout<<k<<" ";  /* əməliyyat  3 */
      k = x&y&&z;
          cout<<k<<" ";  /* əməliyyat 4 */
```

```
        x = 1;
        y = -1;
        k = !x|x;
          cout<<k<<" ";  /* əməliyyat  5 */
        k = -x|x;
          cout<<k<<" ";    /* əməliyya  6  */
        k = x^x;
          cout<<k<<" ";    /* əməliyya  7  */
        x <<= 3;
          cout<<x<<" ";    /* əməliyya  8  */
        y <<= 3;
          cout<<y<<" ";    /* əməliyya  9  */
        y >>= 3;
          cout<<y<<" ";    /* əməliyyat 10 */
}

Proqramın verdiyi nəticə: 3
3 1 1 1 -1 0 8 -8 -1
```

Qeyd: *Biz burada başqa bir hesablama sistemi ilə tanış oluruq. 0 ilə başlayan tam olan sabitlər 8-lik say sisteminin ədədləridir. 8-lik sistemi ilə tam ədədlərlə işləmə çox asandır, siniflər ilə işləyən zaman 8-likdən 2-liyə keçmə asan baş verir. Bu məsələdə 01, 02, 03 ədədləri 1, 2 u 3, ədədlərinə uyğun gəlir, beləliklə 8-lik say sistemində, proqram x, y, z dəyərlərini ikilik say sistemi kimi götürür.*

Kodun şərhi

```
Əməliyyat 1.
   x = 03; y = 02; z =
   01; k = x|y&z;
Əməliyyatların prioritetləri nəticəsində:
k = (x|(y&z));.
Daxili hesablamalar birinci aparılır.
   k = (x|(02&01));
   02 & 01 = 00000010 & 00000001 = 00000000 =
   0 k = (x|00);
   03 | 00 = 00000011 | 00000000 = 00000011 = 03
   03

  Əməliyyat 3.
   x = 03; y = 02; z =
   01; k = x^y&~z;
   k =
   (03^02); 1
Əməliyyat 4.
   x = 03; y = 02; z =
   01; k = x&y&&z;
   k = ((x&y)&&z); k
   = ((03&02)&&z); k
   = (02&&z);
   k = (true&&z);
   k = (&&01);
   k = (true&&true)
   true və ya 1
Əməliyyat 5.
   x = 01; k
   = !x|x;
```

```
    k = ((!x)|x);
    k = ((!true)|x);
    k = (false|01);
    k = (0|01);
    1

Əməliyyat 6.
    x = 01; k
    = -x|x;
    k = ((-x)|x);
    k = (-01|01);
    -01 | 01  = 11111111 | 00000001 =  11111111 = -1
    -1

Əməliyyat  7.
    x = 01; k
    = x^x;
    k = (01^01);
    0

Əməliyyat  8.
    x = 01; x
    <<= 3;
    x = 01<<3;
    01 << 3 = 000000001 << 3 = 00001000 = 8
    x = 8;
    8

Əməliyyat  9.
    y = -01; y
    <<= 3;
    y = -01<<3
    -01 << 3 = 11111111 << 3 =  11111000 =  -8
```

```
    y = -
    8; -8
Əməliyyat 10.
    y = -08; y
    >>= 3;
    y = -08>>3;
    -1
```

Qeyd: *Bəzi hallarda -1 əvəzinə başqa cavab ala bilərik (8191). Bu dəyərin alınması bu cür başa düşülür ki, bəzi kompüterlərdə sürüşmə əməliyyatı zamanı ədədin işarəsi yadda saxlanılmaya bilər. C dilinin hər translyatorları sürüşmə əməliyyatının korrekt olmasına zəmanət vermir, bu səbəbdən daha yaxşı olar ki, 8-ə bölməni adi bölmə əməliyyatı kimi aparaq* ***y=y/8****.*

3. Birləşmələr

Struktur birləşmələrdə istənilən vaxt ancaq bir komponent istifadə oluna bilər. Yazılma şablonu aşağıdakı şəkildə ola bilər:

```
union
  {
   <tip1 adı> <kompanent1>;
   <tip2 adı > <kompanent2>;
         .      .     .
   <tipN adı> <kompanentN>;
  };
```

Strukturun sahəsi yuxarıda qeyd olunmuş ardcıllıqla operativ yaddaşda yerləşdirilir. Birləşmə sahəsi birdən başlayaraq bir birlərinin üzərinə yerləşdirilir.

Birləşmə komponentlərinə keçid, struktura keçid kimi baş verir.

Misal

```
#include <iostream>

using namespace std;

union Test
{
   int a;

   char b;
```

```
}kkk;

void main ()
{
   kkk.a = 65;
   cout<<kkk.a<<" "; //  65 ədədi
   cout<<kkk.b; // A simvolu (həmin ədədə məxsus olan)
}
```

geo_fig birləşməsi və union tipdə obyekti olan misala baxa bilərik[1]:

```
union
{
    int radius; // Dairə.
    int a[2]; // Düzbucaq.
    int b[3]; // Üçbucaq.
} geom_fig;
```

Bu misalda ancaq aktiv komponent hesablanır, yəni elə bir komponent, hansı ki, ən sonda öz dəyərini alır. Məsələn, radius komponentinə dəyər mənimsədəndən sonra, massiv b-yə müraciət etməyin heç bir mənası qalmır.

Qeyd: *Diqqət yetirin ki, bəzi kompüterlərdə bitlərin sahəsi soldan sağa yerləşdirilib, diqərlərində isə sağdan sola. Bu o deməkdir ki, onlar ilə işləmə nə qədər faydalı olsada, əgər məlumatların formatı yüksəkdirsə, sahələrin yerləşdirmə qaydasına nəzarət etmək lazımdır; bu cür hallarnan qarşılaşan proqramlar problem yarada bilər.*

Nəticələr

Birləşmələrin istifadəsi:

- istifadə olunan yaddaşın minimallaşdırılması üçün, hər an ancaq bir obyekt aktiv olur;
- müəyyən tipli əsas obyektin interpretasiyasnda, həmin obyektə başqa tip mənimsədilən zaman.

Beləliklə, verilmiş struktur ilə proqramda bir verilən olacaq, hansıkı özündə bir neçə tipin dəyərini yaddaşda saxlaya bilər.

4. Bit sahələri

Keçən dərsdə biz struktur anlayışını nəzərdən keçirtdik. Strukturun sahəsi ancaq dəyişən olmur, həmçinin bit sahələri də ola bilər. Baxmayaraq ki, proqramlaşdırma dilinin həmin sahələrə heç bir məhdudiyyəti yoxdur, bir tələbdən başqa, onlar maşın kodunun içində yerləşdirilməlidirlər, bit sahələri verilənləri yadda saxlamaq üçün istifadə olunur. (çox vaxt unsigned tipində).

Bit sahələrinin təsviri sahələrin tipindən, adından və iki nöqtədən sonra bitdə verilmiş ölçüdən ibarətdir, məsələn: unsigned status: 6;.

Əgər sahə adı verilməyibsə, onda gizli sahə yaradılır. Əgər bitin sahəsinin ölçüsü 0 verilibsə, o zaman növbəti bitin sahəsi maşın kodunun sərhədindən başlayır.

Misal

```
#include <iostream>
using namespace std;
void Binary(unsigned);
void main()
{
   struct Bits
   {
      unsigned bit1: 3;
      unsigned bit2: 2;
      unsigned bit3: 3;
   } Good;

   Good.bit1 = 4;
   Good.bit2 = 3;
```

```
    Good.bit3 = 6;
    cout<<"Show: "<<Good.bit1<<"
    "; cout<<Good.bit2<<" ";
    cout<<Good.bit3<<"\n\n";
    cout << "Sum: ";
    Binary(Good.bit1 + Good.bit2 + Good.bit3);
}
// Funksiya ekrana A ədədinin ikilik təsvirini verir.
void Binary (unsigned A)
{
     int i,N;
     if(A>255)
                  N = 15;
     else
                  N = 7; for
     (i=N; i >= 0; i--)
     {
          cout<<((A>>i)&1);
                  if(i==8)
                  cout<<" ";
     }
     cout<<"\n\n";
}
```

5. Ev tapşırığı

1. Bit sahələrində kompüterlərin konfiqurasiyası haqqında məlumatın yaddaşda saxlanılmasının proqramını yazmaq. Məsələn: Case AT — 0, ATX — 1; Video onboard — 0, mainboard — 1 və s.
2. Bit sahələrin köməyi ilə kreditlərin verilmə uçotu proqramını yazmaq. Struktur bu sahələrdən ibarətdir: soyad, qrup, kredit (bit sahə) Krediti vermiş və borcu olan tələbələrin siyahısını əlifba sırası ilə nəzərdən keçirtmək.
3. Zamanı (saatlar, dəqiqələr, saniyələr, millisaniyələr) yaddaşda saxlamaq üçün bit sahəsini yaratmaq. Zamanı bit sahəsinə qurub və almaq üçün funksiya yaratmaq.