

Bilgisayar Bilimlerine Giriş-II

-6-

BIL 1002

**Dokuz Eylül Üniversitesi, Fen Fakültesi,
Bilgisayar Bilimleri Bölümü**

Birlikler (Union)

► Birlikler

- Türetilmiş veri tipleridir (Yapılar (struct) gibi).
- Elemanları aynı depolama alanını kullanır.
- **Kullanımı ile değişkenler için hafızayı boş yere işgal etmek yerine ayrılan alanı kullanır.**
- Elemanları her tip olabilir.
- **Depolayabilmek için en az, birliğin en büyük elemanın byte sayısı kadar byte kullanılmalıdır.**
- **Bir anda yalnız bir eleman bu sebeple de yalnızca bir veri tipi kullanılabilir.**

```
union sayi{  
    int x;  
    double y;  
};
```

Birlikler (Union)

- ▶ Birliklerde yapılabilen işlemler
 - Aynı tipte başka bir birliğe atama
 - Adres alma (&)
 - Elemanlarına yapı elemanı operatörü ve yapı gösterici operatörü ile erişme
 - == ve != operatörleri ile karşılaştırılamaz.

```
#include <conio.h>
```

union sayi {

```
int x;
```

```
float y;
```

 $\};$

```
int main()
```

 $\{$

```
union sayi value;
```

```
value.x = 100;
```

```
printf( "%s\n%s\n%s%d\n%s%f\n\n",
```

"Tamsayi uyesine bir sayi koyun",

"ve ikisininide yazdirin.",

```
"int: ", value.x,
```

```
"float:\n", value.y );
```

```
value.y = 100.0;
```

```
printf( "%s\n%s\n%s%d\n%s%f\n",
```

"Ondalik sayi uyesine bir sayi koyun",

"ve ikisininide yazdirin.",

```
"int: ", value.x,
```

```
"float:\n", value.y );
```

```
getch();
```

```
return 0;
```

}

Tamsayi uyesine bir sayi koyunn

ve ikisininide yazdirin.

```
int: 100
```

double:

[illegible][illegible][illegible][illegible]

Öndalik sayi uyesine bir sayi koyun

ve ikisinde yazdırın.

```
int: 0
```

double:

100.000000

1

No	C Struct	C Union
1	Struct tüm üyeleri için bellekte ayrı ayrı yer ayırır.	Union tüm üyeleri için bellekte tek bir ortak yer ayırır. Üyeleri arasındaki en yüksek bellek ihtiyacı bulunan üye kadar bellekte yer ayırır.
2	Struct daha yüksek bir bellek ihtiyacına sahiptir.	Union, struct'ta göre çok daha düşük bellek ihtiyacına sahiptir.
3	Struct'in tüm üyelerine herhangi bir anda erişilebilir.	Union'in yalnızca bir üyesine erişilebilir.
4	<u>Struct:</u> <pre>struct student { int mark; char name[6]; double average; };</pre>	<u>Union:</u> <pre>union student { int mark; char name[6]; double average; };</pre>
5	<p>Yukarıdaki örnek için bellek ihtiyacı:</p> <p>int mark – 2B char name[6] – 6B double average – 8B</p> <p>Total memory allocation = 2+6+8 = 16 Bytes</p>	<p>Yukarıdaki örnek için bellek ihtiyacı:</p> <p>int mark – 2B char name[6] – 6B double average – 8B</p> <p>Total memory allocation = max{2,6,8}=8 Bytes</p>

Bit Operatörleri

Operatör		Tanımlama
&	AND	İki operandın ikisinin de ilgili bitlerinde 1 varsa sonuçtaki bitler 1 yapılır.
	OR	İki operandın ilgili bitlerinden en az biri 1 ise sonuçtaki bitler 1 yapılır.
^	EXCLUSIVE OR	İki operandın ilgili bitlerinden yalnızca biri 1 ise sonuçtaki bitler 1 yapılır.
<<	Sola kaydırma	İlk operandındaki bitleri ikinci operandında belirtilen sayı kadar sola kaydırır.Sağdan itibaren 0 ile doldurur.
>>	Sağa kaydırma	İlk operandındaki bitleri ikinci operandında belirtilen sayı kadar sağa kaydırır.Soldan itibaren yapılacak doldurma makine bağımlıdır.
~	Tümleyen	Tüm 0 bitleri 1,tüm 1 bitleri 0 yapılır.

Bit Operatörleri

AND

Bit1	Bit2	Bit1&Bit2
0	0	0
1	0	0
0	1	0
1	1	1

OR

Bit1	Bit2	Bit1 Bit2
0	0	0
1	0	1
0	1	1
1	1	1

EXCLUSIVE OR

Bit1	Bit2	Bit1^Bit2
0	0	0
1	0	1
0	1	1
1	1	0

Bit Operatörleri

Bit Atama Operatörleri

& =	AND atama operatörü
=	OR atama operatörü
^ =	EXCLUSIVE OR atama operatörü
<< =	Sola kaydırma atama operatörü
>> =	Sağa Kaydırma atama operatörü

Şimdiye kadar görülen operatörlerin öncelikleri ve işleyişleri

Operatör	İşleyiş sırası	Tip
() [] . ->	soldan sağa	en yüksek
++ -- + - (tip) ! & * ~ sizeof	sağdan sola	tekli
* / %	soldan sağa	multiplicative
+ -	soldan sağa	additive
<< >>	soldan sağa	kaydırma
< <= > >=	soldan sağa	karşılaştırma
= = !=	soldan sağa	eşitlik
&	soldan sağa	AND
^	soldan sağa	EXCLUSIVE OR
	soldan sağa	OR
&&	soldan sağa	mantıksal ve
	soldan sağa	mantıksal veya
?:	sağdan sola	koşullu
= += -= *= /= &= = ^= <<= >>=	sağdan sola	atama


```
#include <conio.h>
```

```
void displayBits( unsigned );
```

```
int main()
```

 $\{$

```
unsigned x;
```

```
printf( "Bir tamsayi giriniz: " );
```

```
scanf( "%d", &x );
```

```
displayBits( x );
```

```
return 0;
```

}

```
void displayBits( unsigned value )
```

 $\{$

```
unsigned c, displayMask = 1 << 31;
```

```
printf( "%7u = ", value );
```

```
for ( c = 1; c <= 32; c++ )
```

$$\{$$

```
putchar( value & displayMask ? '1' : '0' );
```

```
displayMask >= 1;
```

}

```
getch();
```

```
putchar( '\n' );
```

}

Bir tamsayı giriniz: 650

[illegible]

```

#include <stdio.h>
#include <conio.h>

void displayBits( unsigned );
int main()
{
    unsigned number1, number2, mask, setBits;

    number1 = 65535;
    mask = 1;
    printf( "Asagidakilerin birlesmesi sonucu:\n" );
    displayBits( number1 );
    displayBits( mask );
    printf( "AND & kullanildiginda\n" );
    displayBits( number1 & mask );

    number1 = 15;
    setBits = 241;
    printf( "\nAsagidakilerin birlesmesi sonucu:\n" );
    displayBits( number1 );
    displayBits( setBits );
    printf( "OR | kullanildiginda\n" );
    displayBits( number1 | setBits );

```

```

Asagidakilerin birlesmesi sonucu:
65535 = 00000000 00000000 11111111 11111111
1 = 00000000 00000000 00000000 00000001
AND & kullanildiginda
1 = 00000000 00000000 00000000 00000001

Asagidakilerin birlesmesi sonucu:
15 = 00000000 00000000 00000000 00001111
241 = 00000000 00000000 00000000 11110001
OR | kullanildiginda
255 = 00000000 00000000 00000000 11111111

Asagidakilerin birlesmesi sonucu
139 = 00000000 00000000 00000000 10001011
199 = 00000000 00000000 00000000 11000111
EXCLUSIVE OR ^ kullanildiginda
76 = 00000000 00000000 00000000 01001100

Asagidakinin bitlerine göre tumleyeni
21845 = 00000000 00000000 01010101 01010101
=
4294945450 = 11111111 11111111 10101010 10101010

```

```

number1 = 139;
    number2 = 199;
    printf( "\nAsagidakilerin birlesmesi sonucu\n" );
    displayBits( number1 );
    displayBits( number2 );
    printf( "EXCLUSIVE OR ^ kullanildiginda\n" );
    displayBits( number1 ^ number2 );

    number1 = 21845;
    printf( "\nAsagidakinin bitlerine göre tumleyeni\n" );
    displayBits( number1 );
    printf( "=\n" );
    displayBits( ~number1 );
    getch();
    return 0;
}

void displayBits( unsigned value )
{
    unsigned c, displayMask = 1 << 31;
    printf( "%7u = ", value );
    for ( c = 1; c <= 32; c++ ) {
        putchar( value & displayMask ? '1' : '0' );
        displayMask >>= 1;
        if ( c % 8 == 0 )
            putchar( ' ' );
    }
    getch();
    putchar( '\n' );
}

```

```

#include <stdio.h>
#include <conio.h>
void displayBits( unsigned );
int main()
{
    unsigned number1 = 960;
    printf( "\nAsagidaki sayiyi\n" );
    displayBits( number1 );
    printf( "8 bit sola kaydirirsak\n " );
    displayBits( number1 << 8 );

    printf( "\nAsagidaki sayiyi\n" );
    displayBits( number1 );
    printf( "8 bit saga kaydirirsak\n " );
    displayBits( number1 >> 8 );
    getch();
    return 0;
}

```

```

void displayBits( unsigned value )
{
    unsigned c, displayMask = 1 << 31;
    printf( "%7u = ", value );
    for ( c = 1; c <= 32; c++ ) {
        putchar( value & displayMask ? '1' : '0' );
        displayMask >>= 1;
        if ( c % 8 == 0 )
            putchar( ' ' );
    }
    getch();
    putchar( '\n' );
}

```

```

Asagidaki sayiyi
  960 = 00000000 00000000 00000011 11000000
8 bit sola kaydirirsak
245760 = 00000000 00000011 11000000 00000000

Asagidaki sayiyi
  960 = 00000000 00000000 00000011 11000000
8 bit saga kaydirirsak
   3 = 00000000 00000000 00000000 00000011

```

Genel Programlama Hataları

- ▶ **OR** operatörü yerine (**|**) mantıksal **VEYA** (**||**) operatörünü kullanmak
- ▶ **AND** operatörü yerine (**&**) mantıksal **VE**(**&&**) operatörünü kullanmak
- ▶ Bir değer kaydırırken eğer sağdaki operand negatif ise ya da sağdaki operandın soldaki operandın bit sayısından büyükse, kaydırma tanımsızdır.

Bit Alanları

C, bir birliğin ya da yapının **unsigned** ya da **int** elemanlarının kaç bit içinde (bit alanı olarak bilinir) depolanacağını belirlememize imkan tanır. Bit alanları, verileri gerekli en az sayıda bit içinde tutarak daha iyi bir hafıza kullanımını sağlar. Bit alanları **int** ya da **unsigned** olarak bildirilir.

```
struct bitKart{  
    unsigned taraf :4;  
    unsigned takim:2;  
    unsigned renk :1;  
};
```

Bu tanımlama 52 kartlık bir desteyi temsil etmek için, 3 **unsigned** bit alanı (**taraf**, **takim** ve **renk**) içerir. Bir bit alanı, **unsigned** ya da **int** bir eleman isminden sonra iki nokta üst üste (:) ve alanın genişliğini belirten bir tamsayı sabiti ile (elemanın depolanacağı bit sayısını belirten bir sabit ile) bildirilir. Genişliği belirten sabit, 0 ile sisteminizde **int** depolamak için kullanılan toplam bit sayısı arasında bir tamsayı olmak zorundadır.

Sayma Sabitleri (Enumerations)

- C, kullanıcı tarafından tanımlanabilen son veri tipi olan *sayma* tipini sunar. Bir sayma, *enum* anahtar kelimesiyle tanıtılır ve tanıtıcılar ile temsil edilen tamsayı sabitlerinin kümesidir. Bu *sayma sabitleri*, değerleri otomatik olarak belirlenen sembolik sabitlerdir. *enum* içindeki değerler aksi belirtilmedikçe 0 ile başlar ve 1 arttırılır.

enum aylar {OCA, SUB, MAR, NIS, MAY, HAZ, TEM, AGU, EYL, EKI, KAS, ARA};

- Bu sayma, tanıtıcıları 0 ile 11 arasında tamsayılar yapar. Ayları 1'den 12'ye kadar saydırmak için

enum aylar {OCA = 1, SUB, MAR, NIS, MAY, HAZ, TEM, AGU, EYL, EKI, KAS, ARA};

```

#include <stdio.h>
#include <conio.h>
enum aylar { OCA = 1, SUB, MAR, NIS, MAY, HAZ,
             TEM, AGU, EYL, EKI, KAS, ARA };

int main()
{
    enum aylar ay;
    const char *ayisim[] = { "Ocak", "Subat", "Mart",
                             "Nisan", "Mayıs", "Haziran", "Temmuz",
                             "Ağustos", "Eylül", "Ekim",
                             "Kasım", "Aralık" };

    printf("Kacinci aydasiniz?");
    scanf("%d",&ay);

```

```

Kacinci aydasiniz?10
Sonbahar mevsimindesiniz

```

```

    if(ay>=MAR && ay<=MAY)
    {
        printf("Ilkbahar mevsimindesiniz");
    }
    else{
        if(ay>=HAZ && ay<=AGU)
        {
            printf("Yaz mevsimindesiniz");
        }
        else{
            if(ay>=EYL && ay<=KAS)
            {
                printf("Sonbahar mevsimindesiniz");
            }
            else
                printf("Kis mevsimidir");
        }
    }
    getch();
    return 0;
}

```

Bilinmeyen Tip Örneği: Struct-Union-Enum bir arada kullanımı

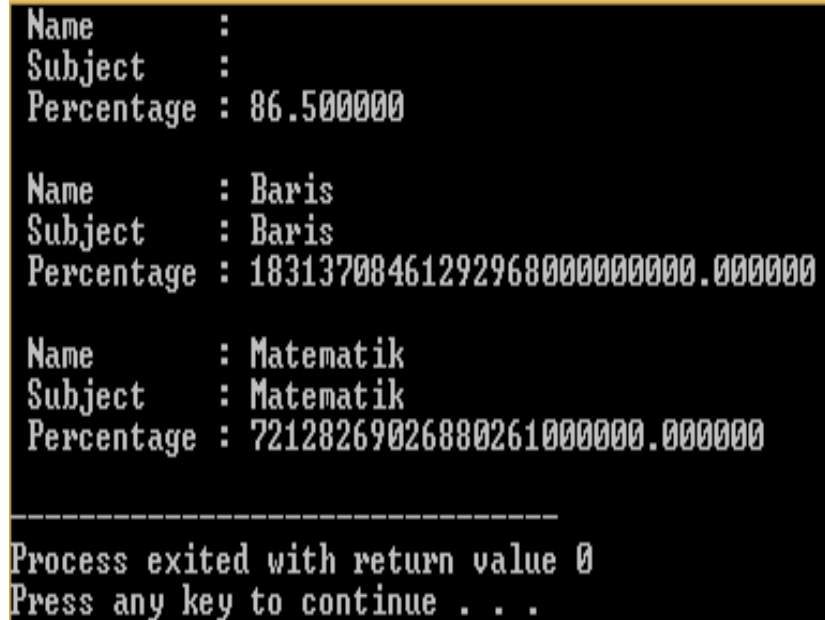
```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
typedef enum { INTEGER, REAL } Type;
typedef struct
{
    Type type;
    union {
        int integer;
        float real;
    } x;
} Value;
Value value_new_integer(int v)
{
    Value val;
    val.type = INTEGER;
    val.x.integer = v;
    return val;
}
Value value_new_double(double v)
{
    Value val;
    val.type = REAL;
    val.x.real = v;
    return val;
}
```

```
int main()
{
    int n=10,i;
    Value *p;
    scanf("%d",&n);
    p=(Value*)malloc(n*sizeof(Value));
    for(i=0;i<n;i++)
    {
        if(i%2==0)
        {
            p[i]=value_new_integer(i);
        }
        else
        {
            p[i]=value_new_double((double)i/2);
        }
    }
    for(i=0;i<n;i++)
    {
        if(p[i].type==INTEGER)
        {
            printf("%d\n",p[i].x.integer);
        }
        else
        {
            printf("%lf\n",p[i].x.real);
        }
    }
}
```


ÖRNEKLER

Soru 1: iki adet string bir adet float değişkeni olan bir union tanımlayınız daha sonra sırası ile tüm değişkenlere ilk değer atayarak, sadece string'lerin birine değer atayarak, diğer string'e değer atayarak ekrana yazdıran bir program yazınız. .

```
#include <stdio.h>
#include <string.h>
union ogr
{
    char name[20];
    char subject[20];
    float percentage;
}rd;
int main()
{
    strcpy(rd.name, "Baris");
    strcpy(rd.subject, "Matematik");
    rd.percentage = 86.50;
    printf(" Name    : %s \n", rd.name);
    printf(" Subject  : %s \n", rd.subject);
    printf(" Percentage : %f \n", rd.percentage);
    printf("\n");
    strcpy(rd.name, "Baris");
    printf(" Name    : %s \n", rd.name);
    printf(" Subject  : %s \n", rd.subject);
    printf(" Percentage : %f \n", rd.percentage);
    printf("\n");
    strcpy(rd.subject, "Matematik");
    printf(" Name    : %s \n", rd.name);
    printf(" Subject  : %s \n", rd.subject);
    printf(" Percentage : %f \n", rd.percentage);
    return 0;
}
```



```
Name      :
Subject    :
Percentage : 86.500000

Name      : Baris
Subject    : Baris
Percentage : 183137084612929680000000000.000000

Name      : Matematik
Subject    : Matematik
Percentage : 72128269026880261000000.000000

-----
Process exited with return value 0
Press any key to continue . . .
```

Soru 2: Bir süpermarkette iki adet ürün için markette ne kadar kaldığını söyleyen bir program yazınız. Bu programda ürünün ismini, fiyatını, tipini ve ne kadar kaldığını içeren bir **struct** yapısı tanımlayınız. Ne kadar kaldığını belirten değişken **union** yapısında olsun mesela farklı birimler için (kg ve adet gibi).

```
#include <string.h>
#include <stdio.h>
typedef union {
    int units;
    float kgs;
} amount ;

typedef struct {
    char selling[15];
    float unitprice;
    int unittype;
    amount howmuch;
} product;

int main() {

    product motosiklet;
    product elma;
    product * myebaystore[2];

    int nitems = 2; int i;

    strcpy(motosiklet.selling,"Dizel Motosiklet");
    motosiklet.unitprice = 5488.00;
    motosiklet.unittype = 1;
    motosiklet.howmuch.units = 4;
```

```
strcpy(elma.selling,"Granny");
elma.unitprice = 0.78;
elma.unittype = 2;
elma.howmuch.kgs = 0.5;
```

```
myebaystore[0] = &motosiklet;
myebaystore[1] = &elma;
```

```
for (i=0; i<nitems; i++) {
    printf("\n%s\n",myebaystore[i]->selling);
    switch (myebaystore[i]->unittype) {
        case 1:
            printf("Elimizde %d adet var.\n",
                myebaystore[i]->howmuch.units);
            break;
        case 2:
            printf("Elimizde %f kg var.\n",
                myebaystore[i]->howmuch.kgs);
            break;
    }
}

return 0;
}
```

```
Dizel Motosiklet
Elimizde 4 adet var.
```

```
Granny
Elimizde 0.500000 kg var.
```

```
-----
Process exited with return value 0
Press any key to continue . . .
```

Soru 3: Girdiğiniz bir tam sayının bitlerinin toplamının tek veya çift olduğuna karar veren bir C programı yazınız.

```
#include <stdio.h>
#include <conio.h>
int bitcift( unsigned );
int main()
{
    unsigned x;
    printf( "Bir tamsayi giriniz: " );
    scanf( "%d", &x );
    printf("Girdiginiz tam sayinin bit sayisi, ");
    if(bitcift(x)) printf("ciftidir.\n");
    else printf("tekdir.\n");

    getch();
    return 0;
}
```

```
int bitcift( unsigned value )
{
    unsigned c, displayMask = 1 << 31,toplam=0;
    for ( c = 1; c <= 32; c++ , displayMask >>= 1 )
    {
        toplam+=value & displayMask;
    }

    return !(toplam%2);
}
```

```
Bir tamsayi giriniz: 3
Girdiginiz tam sayinin bit sayisi, ciftidir.
```

Soru 4: Girdiğiniz iki tam sayıyı bellekte hiç ekstra kullanmadan yerlerini değiştiriniz.

Not: Bit işlemi kullanınız.

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

```
#include <conio.h>
```

```
void xorSwap( int *x, int *y );
```

```
int main()
```

```
{
```

```
    int x,y;
```

```
    printf( "İki tamsayı giriniz: " );
```

```
    scanf( "%d%d", &x,&y );
```

```
    printf("x=%d\n",x,y);
```

```
    xorSwap(&x,&y);
```

```
    printf("\nSwap işlemi yapıldıktan sonra:\n");
```

```
    printf("x=%d\n",x,y);
```

```
    getch();
```

```
    return 0;
```

```
}
```

```
void xorSwap( int *x, int *y ) {
```

```
    if (x != y) {
```

```
        *x ^= *y;
```

```
        *y ^= *x;
```

```
        *x ^= *y;
```

```
    }
```

```
}
```

```
İki tamsayı giriniz: 56 89
x=56
y=89
Swap işlemi yapıldıktan sonra:
x=89
y=56_
```

SON