EE-302 Mikroişlemciler

2. Hafta

Ders Öğretim Üyeleri

Prof. Dr. Mehmet DEMİRTAŞ

C Programlama Dili Tarihçesi

- C programlama dili 1972 yılında AT&T laboratuvarlarında Ken Thompson ve Dennis Ritchie tarafından geliştirilmiştir.
- C dili UNIX işletim sistemini geliştirmek amacıyla B dilinden türetilmiştir.
- Türetildiği yıllarda fazla yaygın olmayan dil, Brian Kernighan ve Dennis Ritchie tarafından yayımlanan “C Programlama Dili” kitabından sonra yaygınlık kazanmıştır.
- C dili günümüzde oldukça yaygın olarak kullanılmaktadır. Öyle ki Windows, GNU/Linux, BSD, Minix gibi işletim sistemlerinin büyük bir çoğunluğu c dili ile yazılmıştır.

C Programlama Dili Tarihçesi

- Peki neden c dili bu kadar çok tercih ediliyor?
  - C, makine dili ile üst seviye diller arasında bir yerdedir. Bu yüzden hem sistem ile ilgili işleri hem de kullanıcıya yakın işleri birlikte yapabilir.
  - Makine diline yakın olduğu için hızlı çalışır.
  - Güçlü esnek ve gelişebilir bir dildir.
  - Çok yaygın olduğu için çok sayıda derleyici ve kütüphaneye sahiptir.
  - Gelişimini tamamlamış bir dildir.
  - Donanımdan bağımsız ve taşınabilir bir dildir.
- Bu kadar etkili olan bir dil sadece Bell laboratuvarlarında kalmadı kısa zamanda yaygınlaşmaya başladı. Doğal olarak geliştiriciler ve kurumlar bu kodları kullanarak birbirinden farklı C dili yarattılar. Bu durum sistem yöneticilerini zor durumda bırakmaya başladı. Ortaya çıkan sorunu çözmek için ABD Ulusal Standartlar Enstitüsü ANSI duruma el koydu ve 1989 yılında ANSI C standardizasyonunu ortaya koydu.

Bir C Programının yapısı

- Her C programı #include direktifi ile başlar. Kullanımı #include <KütüphaneAdı.h> şeklindedir.
- Kütüphaneler programımızda kullanacağımız fonksiyonları içerirler.
- Benzer görevleri olan fonksiyonlar aynı kütüphanede toplanmıştır. Örneğin bir matematik programı geliştiriyorsak math.h adlı kütüphaneye ihtiyacımız vardır.
- Bazı kütüphaneler standarttır , bazıları ise sonradan eklenmiştir. stdio, stdlib, math ... gibi kütüphaneler bütün derleyicilerde bulunur, fakat sonradan geliştirilmiş olan gtk , sdl gibi kütüphaneleri bulamayabilirsiniz.
- Bütün C programlarında bir ana fonksiyon bulunur. bu fonksiyon main() ile belirtilir. Bu ilk çalıştırılan fonksiyondur. Daha sonra veri tipleri ve kendi ürettiğimiz fonksiyonları tanımlarız.
- Bu aşamadan sonra printf, scanf gibi kütüphanemizden aldığımız fonksiyonları kullanabiliriz.

İlk C Programı

- Bir programlama dilinde ilk yazılan programın amacı ekrana "Merhaba Dünya" yazdırmaktır.
- Yalnız bunu C ile yaparken Giriş-Çıkış fonksiyonları hakkında bilgi sahibi olmanız gereklidir. C dili fonksiyonlar üzerine kuruludur
- Ekrana yazı yazdırmak printf() fonksiyonu ile sağlanır ve bu komut stdio.h kütüphanesinde bulunmaktadır.

- **merhaba_dunya.c**

```
#include <stdio.h>
int main () {
    printf ("Merhaba Dünya");    //Ekrana Merhaba Dünya yazdırıldı
    return 0;
      }
```

- Ekran çıktısı

Merhaba Dünya

İlk C Programı

Komut	Açıklama
#include<stdio.h>	Derleme öncesi standart giriş ve çıkış başlıklarını eklemek için kullanılan önişlemci komutudur (Ekrana yazdırmak için (çıkış) gerekli olan printf fonksiyonunun ne işe yaradığını derleyiciye aktaran kütüphanedir)
int main()	Yazılan program yürütülmeye(execute) başladıktan sonra çalışan ilk fonksiyondur (Program yürütülmeye ilk olarak main fonksiyonundan başlar)
{	Main fonksiyonunun başlangıcını gösterir
printf(“Merhaba Dünya”);	Ekrana Merhaba Dünya yazdıran komuttur
return 0;	Bu komut programı sonlandırır ve geri dönüş değeri olarak 0 döndürür
}	Main fonksiyonunun bitişini gösterir

Açıklama Yapmak

- Bir program yazarken o program bölümüyle ilgili açıklayıcı bilgi notu ya da hatırlatıcı eklemek ya da ekibinizdeki kişilere kaynak kodunuz içinde not vermek isteyebilirsiniz. Bu durumda kod içerisine açıklama satırları eklenebilir.
- Tek satırlık açıklamalarda
- // Açıklama metni
- Birden çok satırlık açıklamalarda
- /* Açıklama

  metni */

Değişken Tipleri ve Tanımlamaları

- C değişkenleri, program verilerini bellekte bir isim ile konumlandırmak için kullanılır.
- Değişkenin değerlerini çalışma esnasında değiştirebilirsiniz.
- C programlama dilinde kullanılan temel değişken tipleri char, int, float ve double'dir.
- Ancak bu veri tiplerinin önüne short (kısa), unsigned (işaretsiz), signed (işaretli) ve long (uzun) kelimeleri kullanılarak değişik veri tipleri meydana getirilir.
- Bunlara tip değiştiricileri denir. CCS C'de ise hem değişken tipleri ile hem de sadece tip değiştiricileri ile değişken tipi tanımlaması yapılabilir.

Değişken Tipleri ve Tanımlamaları

Değişken Tipi	Açıklama	En Küçük	En Büyük
int1	1 bit'lik tam sayıyı ifade eder	0	1
int8	8 bit'lik tam sayıyı ifade eder	-128	+128
int	int8 ile aynı anlamdadır	-128	+128
unsigned int	İşaretsiz tam sayı	0	255
int16	16 bit'lik tam sayıyı ifade eder	-32768	+32768
unsigned int16	İşaretsiz 16 bit'lik tam sayı	0	65536
int32	32 bit'lik tam sayıyı ifade eder	-2147483648	+2147483648
char	8 bit'lik karakteri ifade eder	0	255
float	32 bit'lik ondalıklı sayıyı ifade eder		
short	int1 ile aynı anlamdadır		
long	int16 ile aynı anlamdadır		
void	sayının belirli bir tanımının olmadığını belirtir		

Değişken Tipleri ve Tanımlamaları

- int a, b; // a ve b adında 8 bitlik değişkenler tanımlanıyor.
- int8 p; // p adında 8 bitlik bir değişken tanımlanıyor.
- float x; // x adında bir ondalıklı değişken tanımlanıyor.
- char i; // Karakter türünde i adında bir değişken tanımlanıyor.
- int16 bilgi; // i bilgi adında 16 bitlik tamsayı tipinde bir değişken tanımlanıyor.

Değişken Tipleri ve Tanımlamaları

- Değişkenler kullanmak için önce tanımlanması gerekir.
- Değişken tanımlandığında bellekte yer ayrılmaz. Sadece tanımlanmış olur. Değer ataması yapıldıktan sonra bellekte boş bir alana kaydedilir.
- Değişkenlere ilk atama yapıldığında kullanıma hazırdır.

Tip	Syntax
Değişken Tanımlandığında	veriTipi degiskenAdi; Örnek: int x, y, z; char karakter;
Değişkene atama yapıldığında	veriTipi degikenAdi = değer; Örnek: int x = 50; y = 30; char karakter = ‘a’;

C Değişkenlerini isimlendirirken Uyulması Gereken Kurallar

- Değişken adı harf ya da alt çizgi ile başlaması gerekir
- Değişkenler büyük, küçük harfe duyarlıdır
- Değişken isimlendirirken harf ve rakamlar kullanılabilir. Türkçe karakter kullanılamaz.
- Hiçbir özel semboller izin verilmez (alt çizgi '_' hariç)
- Değişken ismi bir C dili kelimesi (keyword) olamaz
- Değişkenlere toplam, boy, _deger seklinde örnekler verilebilir.

NOT: CCS C derleyicisinde varsayılan ayar olarak büyük-küçük harf duyarlılığı kapalıdır. #case komutu ile büyük-küçük harf duyarlılığı açılabilir.

CCS C Derleyicisinde Değişken Tanımlama

- CCS C derleyicisinde sayının önüne hiçbir şey yazılmazsa **desimal** olarak algılanır. **0x** eki sayının önüne getirilirse **heksadesimal** sayı olarak, **0b** eki sayının önüne getirilirse **binary** olarak algılanır. **Karakter** tanımlamaları için, karakterler tek tırnak ' içine yazılarak belirtilir. **String** ifade tanımlanırken ise çift tırnak" "kullanılır.

Data Gösterimi	Açıklama
12	Desimal
012	Oktal
0x12	Heksadesimal
0b00010010	Binary
'x'	Karakter
'**0**10'	Oktal karakter
'**x**A5'	Hex karakter
"abcdef"	String ifade

CCS C Derleyicisinde Değişken Tanımlama

- x=253; //x değişkeni içeriği desimal olarak 253 olur.
- i=0xf3; // i değişkeni içeriği heksadesimal olarak f3'tür.
- data="GAZI"; // data değişkeni içeriği string olarak "GAZI" kelimesidir.
- port=0b10010001; // port değişkeni içeriği binary 10010001 değeridir.

Değişken Faaliyet Alanı ve Ömrü

- C Programlama Dilinde 3 Çeşit Değişkenler Vardır:
  - Lokal Değişkenler
  - Global Değişkenler
  - Environment (Çevresel) Değişkenler

Yerel (Lokal) Değişken

- Yerel değişkenlerin kapsamı sadece tanımlandığı bloğa aittir.
- Bu değişkenler genellikle bir fonksiyonun içinde işlevi vardır ve diğer fonksiyonlar tarafında erişilemez.

Yerel (Lokal) Değişken

```
#include <stdio.h>
#include <stdlib.h>
 void test();
 int main() {
    int a = 5;
    int b = 11;
    printf("a = %d - b = %d\n", a, b);
    test();
    return 0;
}
 void test(){
    int m = 22;
    int n = 18;
    printf("m = %d - n = %d \n", m, n);
    //printf("a = %d - b = %d\n", a, b); expected declaration or statement at end of input Hata verir
}
```

- Çıktı:

a = 5 – b = 11

m = 22 – n = 18

- Yukarıdaki kodda main fonksiyonu içerisinde tanımlanan a ve b değişkenleri diğer fonksiyonlar tarafından erişilemez ve kullanılamaz. Eğer kullanılmak istenirse ‘expected declaration or statemen at end of input ‘ hatası verir.

Global Değişkenler

- Global değişkenlerin kapsamı program boyuncadır. Bu değişkenlere programın herhangi bir yerinde ulaşılabilir.
- Bu değişken main fonksiyonu dışında (main ve diğer alt fonksiyonlar dışında) tanımlanır. Böylece, bu değişken işlevi diğer alt fonksiyonlar tarafından da erişilebilir ve kullanılabilir.

Global Değişkenler

```
#include <stdio.h>
#include <stdlib.h>
void test();
int a = 5;
int b = 11;
 int main() {
    printf("*** main fonksiyonu icerigi baslangic ***\n");
    printf("a = %d - b = %d\n", a, b);
    printf("*** main fonksiyonu icerigi bitis ***\n");
     test();
    return 0;
}
 void test() {
    printf("*** test fonksiyonu icerigi baslangic ***\n");
    printf("a = %d - b = %d\n", a, b);
    printf("*** test fonksiyonu icerigi bitis ***\n");
 }
```

main fonksiyonu üzerinde tanımlanmış a, b değişkenleri global tanımlanmıştır. Bu değişkenlere main fonksiyonu ve diğer alt fonksiyonları tarafından erişilip kullanılabilir.

Çıktı:

```
*** main fonksiyonu icerigi baslangic ***
a = 5 – b = 11
*** main fonksiyonu icerigi bitis ***
*** test fonksiyonu icerigi baslangic ***
a = 5 – b = 11
*** test fonksiyonu icerigi bitis ***
```

C Programlama Dilinde Sabitler

- C sabitleri, normal değişkenler gibidir. Ama bir kez değer atandıktan sonra program tarafından değiştirilemez.
- Sabit değerlerde kullanılır. Bu sayede değiştirilmesine izin verilmez. Örneğin Pi sayısı.
- Sabit değerlerin veri tipleri kısıtlaması yoktur. Her veri tipindeki değerleri sabit olarak tanımlayabilirsiniz.
- C programlama dilinde sabitler 2 şekilde tanımlanabilir
    1. 'const' anahtar kelimesi ile
    2. '#define' önişlemci bildirimi ile

- Örnek syntax;
    - const veriTipi değişkenAdi;
    - const veriTipi *değişkenAdi;

C Programlama Dilinde Sabitler

```
#include <stdio.h>

int main() {

   const int SINIR = 500;
   const float PI = 3.141;
   const char ILKHARF = 'A';
   const char BOLUM[] = "Bilgisayar Muhendisligi";
   const char ALTSATIRAIN = '\n';

   printf("Sinir degeri: %d\n", SINIR);
   printf("Pi degeri: %f\n", PI);
   printf("Alfabenin ilk harfi: %c\n", ILKHARF);
   printf("Bolum adi: %s\n", BOLUM);
   printf("C programlama dili %cCkaynak\n",
ALTSATIRAIN);

   return 0;
}
```

Çıktı:

```
Sinir degeri: 500
Pi degeri: 3.141000
Alfabenin ilk harfi: A
Bolum adi: Bilgisayar Mühendisliği
C programlama dili
Ckaynak
```

C Programlama Dilinde Sabitler

```
#include <stdio.h>

#define SINIR 500
#define PI 3.141
#define ILKHARF 'A'
#define BOLUM "Bilgisayar Muhendisligi"
#define ALTSATIRAIN '\n'

int main() {

   printf("Sinir degeri: %d\n", SINIR);
   printf("Pi degeri: %f\n", PI);
   printf("Alfabenin ilk harfi: %c\n", ILKHARF);
   printf("Bolum adi: %s\n", BOLUM);
   printf("C programlama dili %cCkaynak\n", ALTSATIRAIN);

   return 0;
}
```

Çıktı:

```
Sinir degeri: 500
Pi degeri: 3.141000
Alfabenin ilk harfi: A
Bolum adi: Bilgisayar Muhendisligi
C programlama dili
Ckaynak
```

#define ve const arasındaki farklılık

- Birden çok diziyi belirli bir eleman sayısı ile sınırlandırmak istediğinizde bir sabit değer oluşturup dizileri o eleman sayısı kadar sınırlandırabilirsiniz. Bu sayede kodlar daha dinamik ve düzenlenmesi daha kolay olur. Bu şekilde yazılmak istenilen kodlarda #define sabiti kullanılmalıdır. const sabiti ile dizi sınırlandırılması yapılmaz.

```
#include <stdio.h>

#define SINIR 500

void main() {
    const int SINIR2 = 500;

    int sayilar[SINIR] = {1, 2, 3, 4};
   // int sayilar2[SINIR2] = {1, 2, 3, 4}; Bu şekilde tanımlama yapılmaz. Hata verir

printf("%d", sayilar[0]);
   // printf("%d", sayilar2[0]);  Hata verir

    return 0;
}
```

#define ile tanımlanmış SINIR değişkeni dizinin boyutunu belirtirken kullanabilirsiniz. Ama const ile oluşturulmuş sabit değer dizinin sınırını belirtmek için kullanılamaz (variable-sized object may not be initialized hatası verir).

C Programlama Dili Backslash ‘\’ Sabit Karakterleri

- C dilinde özel anlamı olan bazı karakterler vardır.
- Bu özel anlamı olan fonksiyonlardan faydalanmak için sembollerin önünde BackSlash ‘\’ olmalıdır.

BackSlash ‘\’ Karakterler	Açıklama
\b	Bir önceki karakteri siler (Backspace)
\f	Bir sonraki sayfanın başına geçer (form feed)
\n	Bir alt satıra geçer (newline)
\r	Satır başı yapar (carriage return)
\t	Yatay TAB (Horizontal TAB)
\”	Çift tırnak (“) karakterini ekrana yazmak için
\’	Tek tırnak (‘) karakterini ekrana yazmak için
\\	BackSlash ‘\’ karakterini ekrana yazmak için
\v	Dikey TAB (Vertical TAB)
\a	Uyarı sesi üretir (Alert)
\?	Soru işareti ‘?’ karakterini ekrana yazmak için
\N	Sekizlik tabanda sabit (N sekizlik tabanda bir sabittir.)
\XN	Onaltılık tabanda sabit (N onaltılık tabanda bir sabittir.)

C’de Veri Türü Tanımlama

- typedef ifadesini kullanarak C dilindeki veri türlerini temsil eden kelimeleri (int, char, float, vs.) farklı şekilde tanımlayabilirsiniz. Bu şekilde mevcut bir veri türü için yeni bir isim veya yeni bir veri türü oluşturabilirsiniz. typedef ifadesinin genel yapısı

  typedef eski-isim yeni-isim;

- Yukarıdaki işlem satırı ile oluşturulan yeni-isim ifadesini değişken bildiriminde kullanabilirsiniz.
- Aşağıdaki işlem satırı size bir int değişken tanımlamanız için tms ifadesini kullanma olanağı sağlar:
- typedef int tms;

```
#include <stdio.h>
typedef int tms;

void main (void)
{
  tms id1;
  id1 = 21;
  printf("%d", id1);
}
```

% Sabitleri

- % sabitleri verinin hangi formatta olacağını belirtmek için kullanılır. Herhangi bir birime veri gönderilirken veya herhangi bir birimden veri alınırken, alınacak ya da gönderilecek verinin formatını belirtir. CCS C derleyicisinde özellikle seri iletişim komutlarında ve LCD birimine string ifade gönderme komutlarında bu sabitler kullanılmaktadır.

Kod	Anlamı
%d	char ve int türlerini desimal sistemde
%c	Karakter görüntüsü biçiminde
%x	int türünü Hex sistemde (Harfler küçük yazılır)
%X	int türünü Hex sistemde (Harfler büyük yazılır)
%o	int türünü Octal sistemde
%u	unsigned int türünü desimal sistemde
%ld	long türünü desimal sistemde
%lx	long türünü Heksadesimal sistemde
%f	float ve double türlerini desimal sistemde
%e	float ve double türlerini üstel biçimde
%s	Stringleri karakter olarak

Operatörler

- Operatörler, nesneler veya sabitler üzerinde önceden tanımlanmış bir takım işlemleri yapan atomlardır. Operatörler altı grupta toplanabilir;

1. Aritmetiksel operatörler,
2. Atama operatörleri,
3. İlişkisel operatörler,
4. Mantıksal operatörler,
5. Bit operatörleri,
6. Unary operatörler.

Aritmetik operatörler

- Aritmetik operatörler adının da çağrıştırdığı gibi Toplama, Çıkarma, Çarpma, Bölme gibi basit matematiksel işlemleri yapabilmek için kullanılan operatörlerdir.

Operatör	İşlev	Kullanım
+	Toplama	x+y
–	Çıkarma	x-y
*	Çarpma	x*y
/	Bölme	x/y
%	Mod Alma	x%y
++	Sayıyı bir arttırma	x++ veya ++x
—	Sayıyı bir azaltma	y– veya –y

Aritmetik operatörler

- ++ veya -- operatörleri genellikle döngüler içerisinde sayaç değerlerini her turda bir defa arttırmak için kullanılırlar.
- “x++” ile “++x” arasında ki temel fark şudur; “x++” ifadesi kullanıldığında x değişkeni mevcut değeriyle işleme girer ve işlem bittikten sonra x değişkeninin değeri bir arttırılır, “++x” kullanıldığında ise önce ilk değişkenin değeri bir arttırılır daha sonra işleme girilir. Benzer kullanım -- operatörü içinde geçerlidir.

Aritmetik operatörler

```
#include<stdio.h>

int main() {
   int a = 100, b = 20, toplam, fark, carpim, bolum, mod;
   toplam = a + b;
   fark = a - b;
   carpim = a*b;
   bolum = a / b;
   mod = a % b;
   printf("A ve B toplami : %d\n", toplam);
   printf("A ve B farki : %d\n", fark);
   printf("A ve B carpimi : %d\n", carpim);
   printf("A ve B bolumu : %d\n", bolum);
   printf("A ve B modu : %d\n", mod);
   printf("%d",a++);        // Önce a değişkenini ekrana yazdırıp sonra değerini bir arttırıyor.
   printf("%d",++b);        // Önce b değişkeninin değerini bir attırıp sonra ekrana yazdırıyor.
   return 0;
}
```

Çıktı:

```
A ve B toplamı : 120
A ve B farkı       : 80
A ve B çarpımı : 2000
A ve B bölümü : 5
A ve B modu : 0
100
21
```

Atama Operatörleri

- C Programlama dilinde tanımladığımız değişkenlere bir değer atamak için ”atama” operatörleri kullanılmaktadır.
- Atama operatörlerindeki dikkat edilmesi gereken en önemli nokta atama işleminin sağdan sola doğru olmasıdır. Yani ; (5=a yanlış , a=5 doğrudur)
- Atama operatörleri temel atama operatörü ve aritmetik atama operatörleri şeklinde 2 türdür.

Atama Operatörleri

Operatör	İşlev	Kullanım	Açılım
=	Sağda verilen değeri soldaki değişkene ata	x = y	x = y
+=	Verilen iki sayıyı topla, sonucu soldakine ata.	x+=y	x=x+y
-=	Soldaki sayıdan sağdaki sayıyı çıkar, sonucu soldakine ata.	x-=y	x=x-y
*=	Verilen iki sayıyı çarp, sonucu soldakine ata.	x*=y	x=x*y
/=	Soldaki sayıyı sağdaki sayıya böl, sonucu soldakine ata.	x/=y	x=x/y
%=	Soldaki sayının sağdaki sayıya göre modunu, soldakine ata.	x%=y	x=x%y

Atama Operatörleri

```
#include <stdio.h>
int main(){
   int a, b, x, y, z;
   a=22;
   b=15;
   x=5;
   y=10;
   z=2;
   a+=b;       // a ile b değişkenini toplayıp sonucu a değişkenine atıyor.
   b-=x;       // b değişkeninden x değişkenini çıkarıp sonucu b'ye atıyor.
   x*=y;       // x ve y değişkenlerini çarpıp sonucu x değişkenine atıyor.
   y/=z;       // y değişkenini z değişkenine bölüp sonucu y'ye atıyor.
   printf("%d , %d , %d , %d", a, b, x, y);
   return 0;
}
```

İlişkisel operatörler

- Bu operatörler genellikle if yapıları veya döngülerde kullanılmaktadır. Temel amaç verilen iki değişken veya değişken grubunu belirtilen şarta göre karşılaştırmaktır. Bu karşılaştırmalar ayrı türdeki değişkenler arasında olmalıdır.

Operatör	Kullanım	Açıklama
>	x>y	x y'den büyük mü?
<	x<y	x y'den küçük mü?
>=	x>=y	x y'den büyük ve ya eşit mi?
<=	x<=y	x y'den küçük ya da eşit mi?
==	x==y	x y ye eşit mi?
!=	x!=y	x y'den faklı mı?

İlişkisel operatörler

```
#include <stdio.h>
int main(){
   int a=10, b=11, c=9, d=11, e=9;
   // 0 yanlış, 1 doğru anlamına gelmektedir
   printf("%d < %d = %d", a, b, a<b);
   printf("a = %d > b = %d = %d", a, b, a>b);
   printf("a = %d > c = %d = %d", a, c, a>c);
   printf("a = %d < b = %d = %d", a, b, a<b);
   printf("b = %d == d = %d = %d", b, d, b==d);
   printf("a = %d < b = %d = %d", a, b, a<b);
   printf("c = %d != e = %d = %d", c, e, c!=e);
   printf("c = %d <= e = %d = %d", c, e, c<=e);
   return 0;
}
```

Mantıksal Operatörler

Operatör	İşlev	Kullanım	Açıklama
&&	Mantıksal ve (and)	x&&y	Operatörün her iki tarafındaki değerde de 1 ise sonuç 1'dir. Diğer durumlarda sonuç 0'dır.
||	Mantıksal veya (or)	x||y	Operatörün iki tarafındaki değerlerden herhangi biri ya da ikisi birden 1 ise sonuçta 1'dir. Diğer durumlarda sonuç 0'dır.
!	Mantıksal değil (not)	!x	Operatörün sağındaki değer 1 ise sonuç 0, 0 ise sonuç 1'dir.

x	y	x && y	x || y	x	!x
0	0	0	0	1	0
0	1	0	1	0	1
1	1	1	1		
1	0	0	1		

Mantıksal Operatörler

Örnek	Açıklama
x>3 && y<10	x 3'ten büyük ve y 10'dan küçük ise sonuç 1 (true)
x>3 || y<10	x 3'ten büyük veya y 10'dan küçük ise sonuç 1 (true)
!(x>2)	x 2'den büyük ise 0 değilse 1 (true)

Bit operatörleri

- Sayısal veya karakter değişkenlerin üzerinde bit düzeyinde mantıksal işlem yapan operatörler de vardır.

Operatör	Açıklama	Örnek	Sonuç
&	ve	10 & 25 (00001010 & 00011001)	8 (00001000)
|	veya	10 | 25 (00001010 | 00011001)	27 (00011011)
^	özel veya	10 ^ 25 (00001010 ^ 00011001)	19 (00010011)
~	değil	~10 (00001010)	245 (11110101)
>>	sağa kaydırma	12 >> 3 (00001100 >> 3)	1 (00000001)
<<	sola kaydırma	12 << 3 (00001100 << 3)	96 (01100000)

Kaynakça

- Mikrodenetleyiciler Ders notları, Prof. Dr. Mehmet Demirtaş, Gazi Üniversitesi, 2023.
- Serdar Çiçek, CCS C ile PIC Programlama, Altaş Yayınları, 2007.
- Pic Proje Ekibi, CCS C ile Microchip Pic programlama Klavuzu, Picproje.org 2018.
- Koray Özsoy, Bekir Aksoy, Mikrodenetleyiciler ve programlama, İksad Yayınevi, 2019.