

BİLGİSAYAR BİLİMLERİNE GİRİŞ I

-5-

Programlama Dillerine Giriş

2

- Dilin özellikleri
 - ▣ Alfabetesi
 - ▣ Gramer, yazım kuralları (syntax)
 - ▣ Anlamı (semantics)
- Türkçe ile Japonca'yı karşılaştırınız.
- Programlama dili nedir?
 - ▣ Herhangi bir algoritmayı (çözüm tarifini) ve yapılacak işlemleri bilgisayar tarafından anlaşılıp çalıştırılmak üzere geliştirilmiş, yapay bir dildir.

Programlama Dillerine Giriş

3

- Programlama dillerinin değerlendirme kriterleri:
 - ▣ Readability (Okunabilirlik)
 - ▣ Writability (Yazılabilirlik)
 - ▣ Reliability (Güvenilirlik)
 - ▣ Portability (Taşınabilirlik)
 - ▣ Cost (Maliyet)
 - ▣ Modelling capability (Modelleme yeteneği, becerisi)
 - ▣ Programming Environment (Programlama ortamı, editörü)

Programlama Dillerine Giriş

4

- Temel Programlama Elemanları
 - ▣ Değişkenler ve Sabitler
 - ▣ Operatörler
 - ▣ Koşul Yapıları
 - ▣ Döngü Yapıları
 - ▣ Fonksiyonlar/Prosedürler

5

-

C Programlama Dilinin Tarihçesi

6

- 1970'lerin sonunda, C şu anda geleneksel C olarak bilinen haline geldi.
- Zamanla C'nin yayılması, birbirine benzer ama genellikle uyumsuz, bir çok çeşidinin ortaya çıkmasına sebep oldu.
- 1983 yılında, *American National Standards Committee*'nin bilgisayar ve bilgi işlem komitesi tarafından C'nin sistem bağımsız bir tanımı yapıldı.
- 1989 yılında bu standart onaylandı ve 1999 yılında tekrar gözden geçirildi.

C Programlama Dili Tercih Nedeni

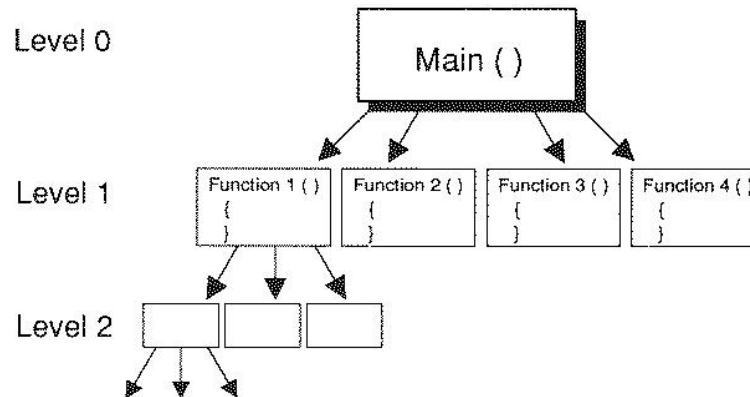
7

- En **popüler** dillerden birisidir.
- Güçlü ve **esnek** bir dildir. C ile işletim sistemi veya derleyici yazabilir, kelime işlemciler oluşturabilir veya grafikler çizebilirsiniz.
- Yazılım geliştirme ortamları oldukça fazladır.
- Özel komut ve veri tipi tanımlamasına izin verir.
- **Taşınabilir** bir dildir.
- Gelişimini tamamlamış ve **standardı oluşmuş** bir dildir.
- **Yapısal** bir dildir. C kodları fonksiyon olarak adlandırılan alt programlardan oluşmuştur.

Standart C Kütüphanesi

8

- C programları «fonksiyon» adı verilen parçalardan yada modüllerden oluşur.
- Fonksiyonlar C «bloklarından» oluşur.
- Her fonksiyon/blok bir veya daha fazla «deyimi» içerir.
- Her bir deyim program çalıştırıldığında belirli bir eylemi yerine getirir. Deyimler işlemleri yerine getiren komutlardır.

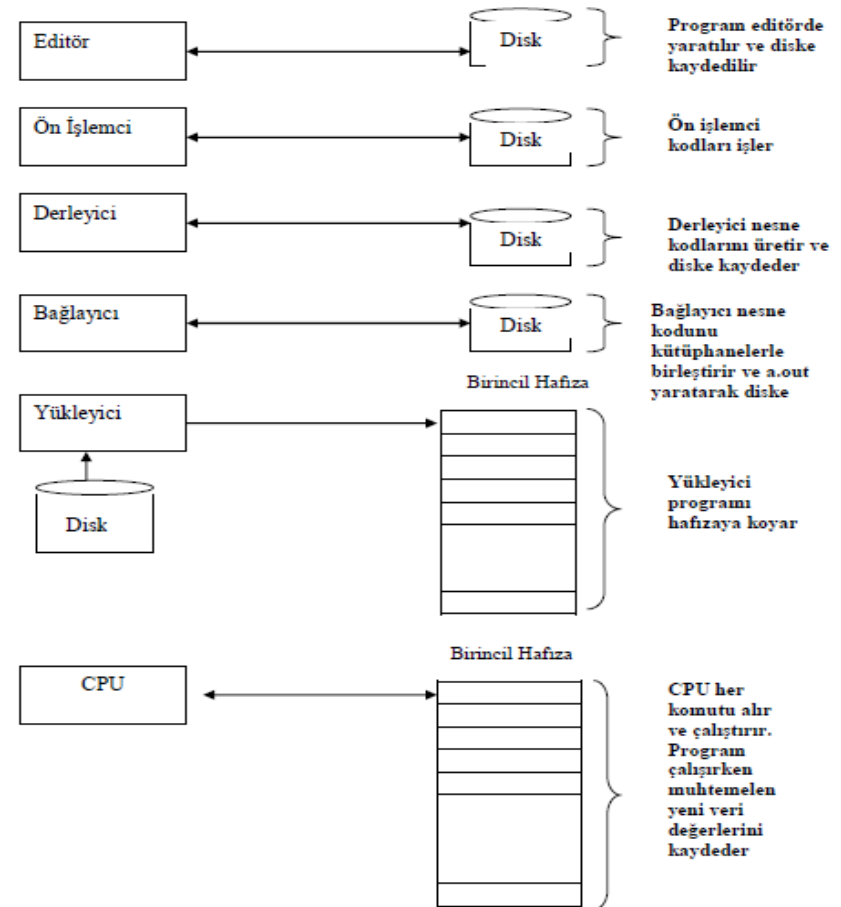


C Programı Geliştirme Ortamının Temelleri

9

□ Tipik olarak bir C programı çalışmadan önce altı safhadan geçer.

- Yazım(Edit)
- Ön işleme(Preprocess)
- Derleme(Compile)
- Bağlama(Link)
- Yükleme(Load)
- Çalıştırma(Execute)



C Dilinin Temel Yazım Özellikleri

10

- Program yazımını bloklar halinde olur.
- Bloklar, { } parantezleri ile oluşturulur.
- Komutlar aynı veya alt alta satırlara yazılabilir.
- Tüm komutlar, noktalı virgül (;) ile biter.
- Yalnız blok başlatan ifadelerden sonra noktalı virgül kullanılmaz.
- Programda kullanılan tüm değişkenler ve veri tipleri bildirilir.
- Programda kullanılacak olan komutların bulunduğu kütüphaneler aktifleştirilir/çağırılır.

C Program Yapısı

11

Ön işlemci Direktifleri
(Preprocessor Directives)

Genel Tanımlamalar
(Global Declarations)

```
main ( )
```

```
{
```

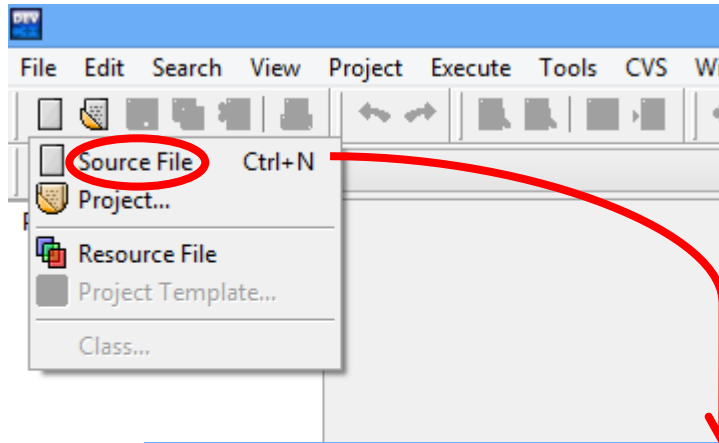
Yerel Tanımlamalar (Local Declarations)

Deyimler ve İfadeler (Statements)

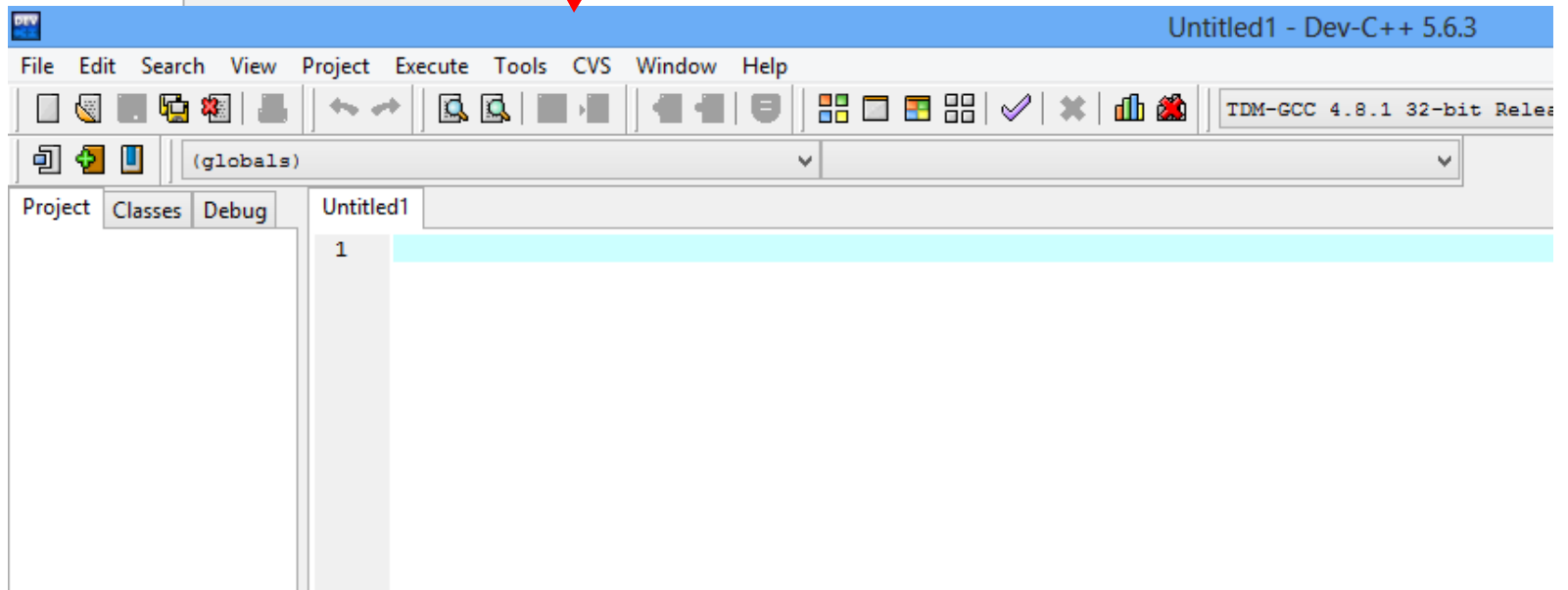
```
}
```

C File Açma

12



File → Source File



İlk Programım

13

```
/* C ile ilk program*/  
#include <stdio.h>  
void main()  
{  
    printf("Merhaba Dünya!");  
}
```

İlk Programım

14

```
/* C ile ilk program*/
```

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

```
void main()
```

```
{
```

```
    printf("Merhaba Dünya!");
```

```
}
```

Ekran görüntüsü;

Merhaba Dünya!

/* ve */ arasına yorum yazılır, derleyici görmezden gelir.

#include C önışlemcisine bir emir göndermektedir. Bu satır, önışlemciye standart giriş/çıkış öncü dosyası(stdio.h) içeriğinin programa eklenmesini söyler.

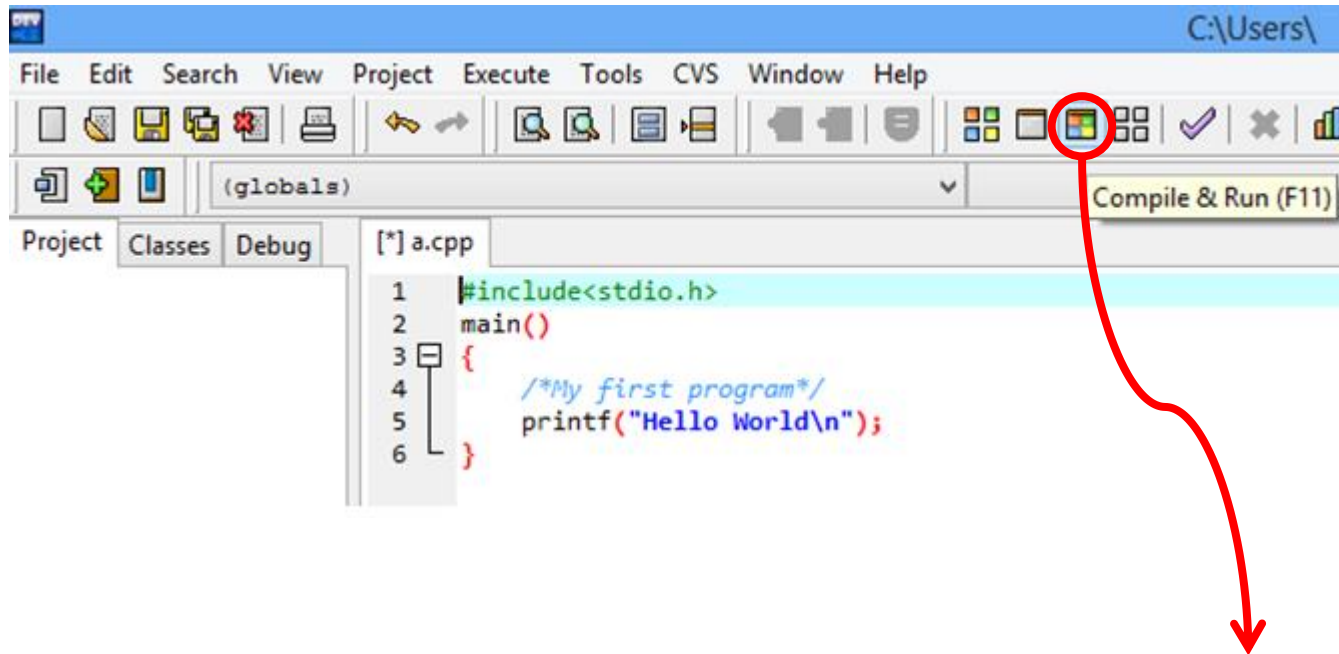
void main() her C programının bir parçasıdır. C programları bir veya birden fazla fonksiyon içerebilir ancak bunlardan biri mutlaka main olmalıdır. C'de her program main fonksiyonunu çalıştırarak başlar.

Küme parantezi, { , her fonksiyonun gövdesinin başına yazılır. }, küme parantezi ise sonuna yazılmalıdır. Bu iki parantez arasında kalan program parçacığına blok denir.

printf konsola yazdırma işlemini gerçekleştirir.

Compile→Run (F11)

15



```
Hello World

-----
Process exited after 0.008779 seconds with return value 0
Press any key to continue . . . _
```

Yorum Satırı

16

□ Tekli ve Çoklu Yorum Satırı

- ▣ // Tek satırda yorum yapılacağı zaman kullanılır.
- ▣ /* Çoklu yorum satırı * bir satırla açıklama yapamayacağımız zaman * kullandığımız bir yöntemdir.
*/

C Programlama Dili Elemanları

17

- Anahtar Sözcükler
- Veri Türleri
- Değişkenler
- Sabitler
- Operatörler

C Anahtar Kelimeleri

18

Anahtar Kelimeler (Keywords)

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

C Veri Türleri

19

- **Veri tipi (data type)** program içinde kullanılacak değişken, sabit, fonksiyon isimleri gibi tanımlayıcıların tipini, yani bellekte ayrılacak bölgenin büyüklüğünü, belirlemek için kullanılır.
- Bir programcı, bir programlama dilinde ilk olarak öğrenmesi gereken, o dile ait veri tipleridir. Çünkü bu, programcının kullanacağı değişkenlerin ve sabitlerin sınırlarını belirler.

C Veri Türleri

20

- C programlama dilinde 5 tane temel veri tipi bulunmaktadır.
 - ▣ **char**: karakter veriler
 - ▣ **int**: tamsayı veriler
 - ▣ **float**: tek duyarlıklı kayan noktalı sayılar
 - ▣ **double**: Çift duyarlıklı kayan noktalı sayılar
 - ▣ **void**: Değer içermeyen verilerdir.

C Değişkenleri

21

- Değişken, program içinde kullanılan değerlere bellek üzerinde açılan alanlardır. Bu alanlar bir **değişken ismi** ile anılır.
- Değişken isimlendirilmeleri, tanımlayıcı kurallarına uygun biçimde yapılmalıdır.
- C'de tüm değişkenler kullanılmadan önce programa bildirilmelidir.
- Bu bildirim esnasında, değişkenin veri türü belirlenir.
- **Örnek:**

```
veri_türü değişken_adı;
```

```
int sayac;
```

Örnek: C Değişkenleri

22

□ Örnekler

- **int** x;

- **int** x1, y1, z1;

- **long** d, d1;

- **char** c;

- **char** c1, c2, c3;

- **float** a;

- **float** a1, a2, a3;

C Sabitleri

23

- Sabit bildirimi, başlangıç değeri verilen değişken bildirimi gibi yapılır.
- Ancak, veri tipinin önüne **const** anahtar sözcüğü konmalıdır.
- Sabit içerikleri **program boyunca değiştirilemez**. Yalnızca kullanılabilir.
- Genellikle, sabit olarak bildirilen değişken isimleri **büyük harflerle**, diğer değişken isimlerinin ise küçük harflerle yazılması (gösterilmesi) C programcıları tarafından **geleneksel** hale gelmiştir.

Örnek: C Sabitleri

24

□ Örnekler:

- `const float PI=3.142857;`
- `const double NOT=12345.8596235489;`
- `const int EOF=-1;`
- `const char metin[]="devam etmek için bir tuşa basın...";`

Tam Sayılar - Integer

25

- Tam sayıları ifade eder
 - ▣ Hem negatif hem pozitif tam sayılar
- C de tam sayıların (integer) ifade tarzı: int
- Örnek:

```
int toplam;    /* işaretli integer */  
  
toplam = 100;  /* pozitif olabilir */  
toplam = -20; /* negatif olabilir */
```

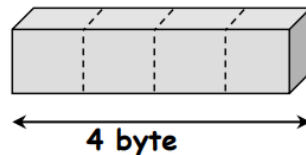
```
int toplam = 32000; /* kodlama sırasında */  
                  /* ilk değer verilebilir */
```

Tam Sayılar – Integer (Devam)

26

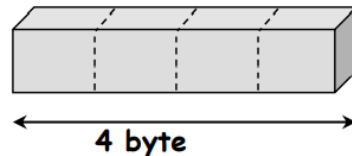
- Integer niteleyicileri: long, short, veya unsigned
- Integer değişkenlerin niteleyicilerine göre büyüklükleri değişir.
- Varsayılan integer büyüklüğü makine/işletim sistemine bağlıdır

int



-2.147.483.648 den 2.147.483.647 e kadar (toplam 4.294.967.296 adet sayı)

**unsigned
int**



0 dan 4,294,967,295 e kadar (toplam 4,294,967,296 adet sayı)

Virgüllü Sayılar - float

27

- Gerçek sayıları ifade eder (virgüllü kısmıyla)
 - ▣ Pozitif ve negatif olabilir
- C de virgüllü sayıların ifade tarzı: float
- Örnek:

```
float f;
```

```
f = 0.12;    /* pozitif olabilir */
```

```
f = -245.56; /* negatif olabilir */
```

```
float f = 4.567; /* kodlama sırasında      */  
                /* ilk değer verilebilir */
```

Daha Uzun ve Çok Hassas Virgüllü Sayılar- double

28

- Standart "double precision floating point" (gerçek) sayılardır.
 - ▣ float gibi, fakat çok daha büyük ve hassastır.
- C deki ifade tarzı: double
- Örnek:

```
double d;
```

```
d = 3.12E+5;    /* 312000.0 */
```

```
d = -45.678;    /* negatif */
```

```
double d = 4.567; /* ilk değer ataması */
```

Karakter - char

29

- Bir tek karakteri ifade eder
 - ▣ Karakterler
 - Alfabedeki büyük ve küçük harfler
 - 0 dan 9 a kadarki 10 numara
 - Özel semboller örneğin
 - `+#@1/2%&$.*?!£'=-:/^*^(){}[]~;,,<>`
- Karakterler tırnak işareti arasında kullanılır
 - ▣ örneğin. 'A'
- C deki kullanım tarzı: char

```
char c;
```

```
c = 'A'; /* A Harfi */
```

```
c = '9'; /* 9 rakamı*/
```

```
char c = 'c'; /* ilk değer verme */
```

Karakter (devam)

30

- Aslında karakterler 1 byte lık doğal sayıları ifade eder
 - ▣ char tipi değişkenler hafızada 1 byte yer tutar
- Karakterlerin (char değişkenleri) ASCII tablosundaki değerleri...
 - ▣ 'A' nın ASCII değeri 65
 - ▣ 'B' nın ASCII değeri 66
 - ▣ '0' ın ASCII değeri 48
 - ▣ '1' in ASCII değeri 49
 - ▣ '5' in ASCII değeri 53
 - ▣ <http://www.asciitable.com/>

```
#include<stdio.h>
main()
{
    char k='5';
    int  a=k-'1';
    printf("k = %c \n", k);
    printf("a = %d", a );
}
```

```
k = 5
a = 4
```

ASCII Tablosu

31

Decimal Hex Char			Decimal Hex Char			Decimal Hex Char			Decimal Hex Char		
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Veri Tipleri ve Özellikleri

32

□ C'de veri tipleri

Veri Tipi	Açıklama	Bellekte işgal ettiği boyut (bayt)	Alt sınır	Üst sınır
char	Tek bir karakter veya küçük tamsayı için	1	-128	127
unsigned char			0	255
short int	Kısa tamsayı için	2	-32,768	32,767
unsigned short int			0	65,535
int	Tamsayı için	4	-2,147,483,648	2,147,483,647
unsigned int			0	4,294,967,295
long int	Uzun tamsayı için	8	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
unsigned long int			0	18,446,744,073,709,551,615
float	Tek duyarlı gerçel sayı için (7 basamak)	4	-3.4e +/- 38	+3.4e +/- 38
double	Çift duyarlı gerçel sayı için (15 basamak)	8	-1.7e +/- 308	+1.7e +/- 308

Ön İşlemci Direktifleri

33

- Başlık dosyaları, derleyicinin kütüphane fonksiyonu çağrılarının doğru yapılıp yapılmadığını anlamasında yardımcı olan bilgiler içerir.
- ANSI C'deki standart başlık dosyaları aşağıdaki gibidir:

❖ assert.h

❖ ctype.h

❖ errno.h

❖ float.h

❖ limits.h

❖ locale.h

❖ math.h

❖ setjmp.h

❖ signal.h

❖ stdarg.h

❖ stddef.h

❖ stdio.h

❖ stdlib.h

❖ string.h

❖ time.h

Escape Sequence – Kaçış Dizisi

34

- Ters slash (\) ve bir karakterden oluşur. Derleyiciye sonraki karakterin normal olarak algılanması işaretini verir.
- Sık kullanılanlar
 - ▣ \n sonraki satıra geç
 - ▣ \t sonraki sekmeye geç
 - ▣ \r satır başına alır
 - ▣ \\ ters slash karakteri –
 - ▣ \' tek tırnak
 - ▣ \" çift tırnak

Format Belirteçleri

35

Belirleyici	Biçim
%d, %i	Tamsayı (Decimal)
%u	İşaretsiz Tamsayı (Unsigned)
%f	Kayan Noktalı Sayı (Float)
%c	Karakter (Char)
%o	8 Tabanında Sayı (Octal)
%x, %X	16 Tabanında Sayı(Hexadecimal)
%e	Üssel Gösterim (Exponential)
%s	Karakter Dizisi (String)
%l, %h	Long ve short ön eki

main() Fonksiyonu

36

- Programın özel bir fonksiyonudur; ana program anlamındadır.
- C diliyle yazılmış bir program yürütülmeye başlandığında ilk bu fonksiyon çağrılır.
- Programın yürütülmesi bu fonksiyondan başlar.

Input/Output Fonksiyonları

37

- I/O fonksiyonları standart input/output C Kütüphanesinde tanımlanmış
 - ▣ `stdio.h`

- Klavye Input
 - ▣ `scanf` -- Genel Formatlanmış input
 - ▣ `getchar` -- tek bir karakter okur

- Monitör (Ekran) Output
 - ▣ `printf` -- Genel Formatlanmış output
 - ▣ `putchar` -- tek bir char (karakter) yazar

printf Fonksiyonu

38

- Ekrana veriyi biçimlendirerek yazabilen bir fonksiyondur.

printf("biçim ifadesi", değişkenler);

- Çift tırnak arasında yer alan ‘biçim ifadesi’ genel olarak üç kısımdan oluşur.
 - ▣ Açıklama kısmı
 - ▣ Biçim kısmı
 - ▣ Kontrol/çıkış Kısmı

Örnek

39

```
int numara = 7;
```

```
printf("%d nin iki katı= %d \n", numara, 2*numara);
```

Biçim kısmı

İfade kısmı

printf Fonksiyonu

40

- **Açıklama:** Çift tırnaklar arasında verilip ekrana doğrudan yazılır.

printf("Ankara");

- **Biçim:** % sembolüyle başlayan ve çıkış biçiminin belirlendiği kısımdır.

printf("Sonuc: %d ", x);

- **.precision:** maksimum kaç karakterde gösterileceğini belirtir.

printf("Sonuc: %.2lf ", y);

printf Fonksiyon Örnekleri

41

```
double fp = 251.7366;  
int i = 25;  
printf("Reel sayi: %.2lf \n", fp);  
printf("Saga yaslanilmis integer: %10d \n", i);
```

Çıktı:

```
Reel sayi: 251.74  
Saga yaslanilmis integer :           25
```

printf Örnekler

42

```
printf("%.5f\n", 300.0123456789);  
printf("%.14lf\n", 300.01234567890123456789);
```

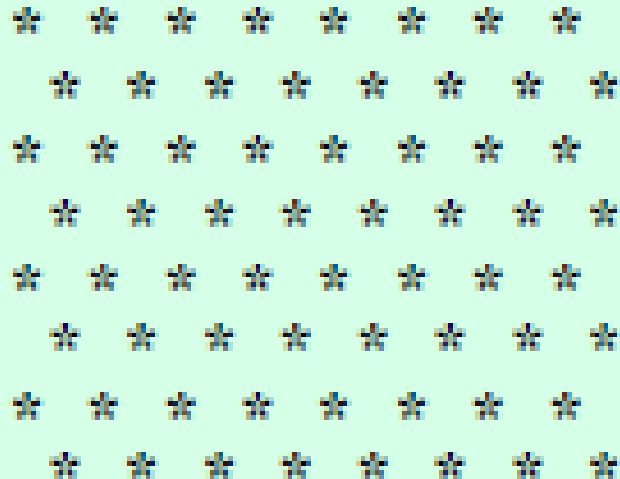
300.01235

300.01234567890123

Örnek

43

Aşağıdaki deseni sekiz **printf** ifadesiyle ekrana yazdıran bir program yazınız. Daha sonra aynısını, kullanabileceğiniz en az **printf** ifadesiyle yazınız.



```

* * * * *
 * * * * *
* * * * *
 * * * * *
* * * * *
 * * * * *
* * * * *
```

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

```
main()
```

```
{
```

```
    printf( "Sekiz printf() ifadesi ile: \n" );
```

```
    printf( "* * * * * *\n" );
```

```
    printf( " * * * * * *\n" );
```

```
    printf( "* * * * * *\n" );
```

```
    printf( " * * * * * *\n" );
```

```
    printf( "* * * * * *\n" );
```

```
    printf( " * * * * * *\n" );
```

```
    printf( "* * * * * *\n" );
```

```
    printf( " * * * * * *\n" );
```

```
    printf( "\nSimdi ise bir printf() ifadesi ile: \n" );
```

```
    printf( "* * * * * *\n * * * * * *\n"
```

```
        "* * * * * *\n * * * * * *\n"
```

```
        "* * * * * *\n * * * * * *\n"
```

```
        "* * * * * *\n * * * * * *\n" );
```

```
}
```



```
Sekiz printf() ifadesi ile:
* * * * * *
 * * * * * *
* * * * * *
 * * * * * *
* * * * * *
 * * * * * *
* * * * * *
 * * * * * *

Simdi ise bir printf() ifadesi ile:
* * * * * *
 * * * * * *
* * * * * *
 * * * * * *
* * * * * *
 * * * * * *
* * * * * *
 * * * * * *
```

scanf Fonksiyonu

45

- Klavyeden belirtilen değişkene veri aktarılmasını sağlar.

scanf(" biçim ifadesi ", &değişkenler listesi);

- Buradaki "biçim ifadesi" veri girişinin hangi biçimde olacağını; "değişkenler (adres) listesi" de verilerin aktarılacağı değişkenleri belirtir.

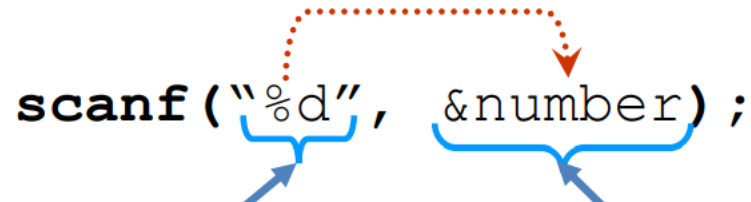
Örnek

46

```
int number;
```

```
printf("Bir integer girin: ");
```

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



Biçim kısmı

Değişken adresi

Örnek: Girilen iki sayıyı ekrana yazdıran program tasarlayınız.

47

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

```
main()
```

```
{
```

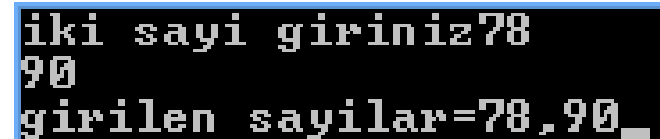
```
    int a,b;
```

```
    printf("iki sayi giriniz");
```

```
    scanf("%d %d",&a,&b);
```

```
    printf("girilen sayilar = %d, %d", a, b);
```

```
}
```



```
iki sayi giriniz78
90
girilen sayilar=78,90_
```

Alıştırmalar

48

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

```
void main()
```

```
{
```

```
    int x;    // "int" tipinde "x" adında bir değişken tanımla.
```

```
    x = 65;   // x değişkenine 65 değerini ata.
```

```
    printf ("%d", x); // x'in değerini ekrana yazdır.
```

```
}
```


Alıştırmalar

49

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

```
void main()
```

```
{
```

```
    int x;    // "int" tipinde "x" adında bir değişken tanımla.
```

```
    x = 65;   // x değişkenine 65 değerini ata.
```

```
    printf ("x in degeri: %d",x); // x'in değerini ekrana yazdır.
```

```
}
```

Alıştırmalar

50

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

```
void main()
```

```
{
```

```
    int x;
```

```
    x = 29;
```

```
    printf ("Onluk tabanda = \t%d\n", x);
```

```
    printf ("Sekizlik tabanda = \t%o\n", x);
```

```
    printf ("Onaltılık tabanda = \t%X\n", x);
```

```
}
```

```
Onluk tabanda =      29
Sekizlik tabanda =   35
Onaltılık tabanda =  1D
```

Alıştırmalar

51

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

```
void main()
```

```
{
```

```
    int x;
```

```
    printf ("Bir tamsayı girin.\n");
```

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

```
    printf ("Onluk tabanda = \t%d\n", x);
```

```
    printf ("Sekizlik tabanda = \t%o\n", x);
```

```
    printf ("Onaltılık tabanda = \t%X\n", x);
```

```
}
```

```
Bir tamsayı girin.
```

```
29
```

```
Onluk tabanda =          29
```

```
Sekizlik tabanda =       35
```

```
Onaltılık tabanda =     1D
```

Alıştırmalar

52

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

```
void main()
```

```
{
```

```
    float x;
```

```
    printf ("Ondalıkli sayı girin(nokta ile).\n");
```

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

```
    printf ("Girilen Sayı = %f \n", x);
```

```
}
```

```
Ondalikli sayi girin(nokta ile).  
12.32  
Girilen Sayi = 12.320000
```

C Operatörleri

53

- Operatörler, değişkenler veya sabitler üzerinde matematiksel ve karşılaştırma işlemlerini yapan simgelerdir. Yani bir operatör bir veya daha fazla değişken üzerinde işlem yapan semboldür.
- C programlama dilinde 4 tip operatör bulunmaktadır.
 - ▣ Aritmetik Operatörler
 - ▣ Atama Operatörleri
 - ▣ Karşılaştırma Operatörleri
 - ▣ Mantıksal Operatörler

C Operatörleri (Aritmetik Operatörler)

54

Operatör	Açıklama	Örnek	Anlamı
+	toplama	$x + y$	x ve y nin toplamı
-	çıkarma	$x - y$	x ve y nin farkı
*	carpma	$x * y$	x ve y nin çarpımı
/	bölme	x / y	x ve y nin oranı
%	mod alma	$x \% y$	x / y den kalan sayı

C Operatörleri (Atama Operatörleri)

55

Operatör	Açıklama	Örnek	Anlamı
=	atama	x = 7;	x = 7;
+=	ekleyerek atama	x += 3	x = x + 3
-=	eksilterek atama	x -= 5	x = x - 5
*=	çarparak atama	x *= 4	x = x * 4
/=	bölerek atama	x /= 2	x = x / 2
%=	bölüp, kalanını atama	x %= 9	x = x % 9
++	bir arttırma	x++ veya ++x	x = x + 1
--	bir azaltma	x-- veya --x	x = x - 1

C Operatörleri (Atama Operatörleri)

56

Örnek	Anlamı
$x = y++;$	y'nin değeri önce x'e aktarılır sonra bir arttırılır. $x = y;$ $y = y + 1;$
$x = ++y;$	y'nin değeri önce bir arttırılır sonra x'e aktarılır . $y = y + 1;$ $x = y;$
$x = y--;$	y'nin değeri önce x'e aktarılır sonra bir azaltılır. $x = y;$ $y = y - 1;$
$x = --y;$	y'nin değeri önce bir azaltılır sonra x'e aktarılır . $y = y - 1;$ $x = y;$

Operatörler

57

□ Karşılaştırma Operatörleri

Operatör	Açıklama	Örnek	Anlamı
>	büyüktür	$x > y$	x, y'den büyük mü?
<	küçüktür	$x < y$	x, y'den küçük mü?
==	eşittir	$x == y$	x, y'ye eşit mi?
>=	büyük-eşittir	$x >= y$	x, y'den büyük veya eşit mi?
<=	küçük-eşittir	$x <= y$	x, y'den küçük veya eşit mi?
!=	eşit değil	$x != y$	x, y'den farklı mı?

Operatörler

58

□ Mantıksal Operatörler

Operatör	Açıklama	Örnek	Anlamı
&&	mantıksal VE	$x > 5 \ \&\& \ x < y$	x, 5'den büyük VE x, y'den küçük mü?
 	mantıksal VEYA	$x > 5 \ \ x < y$	x, 5'den büyük VEYA x, y'den küçük mü?
!	mantıksal DEĞİL	$!(x > 5)$	x, 5'den büyük değilse (x, 5'den küçük VEYA 5'e eşitse)

Örnek

59

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

```
main() {
```

```
    int a = 20, b = 10, c = 15, d = 5, e;
```

```
    e = (a + b) * c / d;    // ( 30 * 15 ) / 5
```

```
    printf("değer: %d\n", e );
```

```
    e = ((a + b) * c) / d;    // (30 * 15 ) / 5
```

```
    printf(" değer: %d\n" , e );
```

```
    e = (a + b) * (c / d);    // (30) * (15/5)
```

```
    printf(" değer: %d\n", e );
```

```
    e = a + (b * c) / d;    // 20 + (150/5)
```

```
    printf(" değer: %d\n" , e );
```

```
}
```

Örnek

60

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

```
main()
```

```
{
```

```
    int m = 7, j = 7, k, z;
```

```
    k = m++;                                /* k = m, m = m + 1 */
```

```
    z = ++j;                                /* j = j + 1, z = j */
```

```
    printf(" %d %d %d %d", m, j, k, z);
```

```
    getch();
```

```
}
```

8 8 7 8

Uygulama 1

61

Klavyeden girilen 2 tamsayının

- ▣ Toplamını
- ▣ Farkını
- ▣ Çarpımını
- ▣ Bölümünü
- ▣ Modunu

bulup yazdıran C programını yazınız.

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

```
main()
```

```
{
```

```
    int x;
```

```
    int y;
```

```
    printf( "Iki Sayi Giriniz: ");
```

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

```
    printf( "Toplam: %d\n", x + y );
```

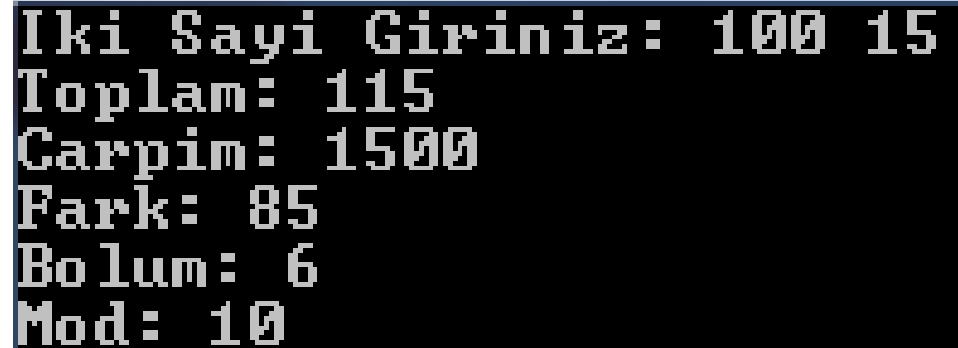
```
    printf( "Carpim: %d\n", x * y );
```

```
    printf( "Fark: %d\n", x - y );
```

```
    printf( "Bolum: %f\n", (float)x / y );
```

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

```
}
```



```
Iki Sayi Giriniz: 100 15
Toplam: 115
Carpim: 1500
Fark: 85
Bolum: 6
Mod: 10
```

Uygulama 2

63

- 1'den 4'e kadar olan tamsayıları ekrana tek satırda görülecek şekilde yazdıran bir programı aşağıdaki metotları kullanarak yazınız.
- Bir printf ifadesi kullanarak ve hiç format belirteci kullanmadan.
- Bir printf ifadesi ve dört format belirteci kullanarak.
- Dört printf ifadesi kullanarak.

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

```
main()
```

```
{
```

```
    printf( "1 2 3 4\n\n" );    /* bir printf ifadesi ile */
```

```
    printf( "%d %d %d %d\n\n", 1, 2, 3, 4 );    /* bir printf ifadesi + 4 format  
                                                  belirteci ile */
```

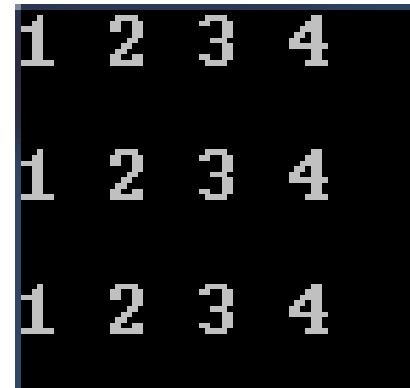
```
    printf( "1 " );    /* 4 printf ifadesi ile */
```

```
    printf( "2 " );
```

```
    printf( "3 " );
```

```
    printf( "4\n" );
```

```
}
```



1	2	3	4
1	2	3	4
1	2	3	4

Uygulama 3

65

Kullanıcıdan bir çemberin yarıçapını alan ve bu çemberin

- ▣ çapını,
- ▣ çevresini,
- ▣ alanını

hesaplayan bir program yazınız.

π için 3.14159 değerini kullanın.

Bütün hesaplamalarınızı kullandığınız printf ifadeleri içinde yaptırın. %f dönüşüm belirtecini kullanın.

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

```
main()
```

```
{
```

```
    int yc;
```

```
    printf( "Yaricapi giriniz: " );
```

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

```
    printf( "\nCap: %d\n", 2 * yc );
```

```
    printf( "Cevre: %f\n", 2 * 3.14159 * yc );
```

```
    printf( "Alan: %f\n", 3.14159 * yc * yc );
```

```
}
```

```
Yaricapi giriniz: 5
```

```
Cap: 10
```

```
Cevre: 31.415900
```

```
Alan: 78.539750
```

Uygulama 4

67

- Beş basamaklı bir sayı girişi yapılan,
 - ▣ bu sayıyı ayrı ayrı basamaklarına ayıran
 - ▣ her basamak arasına üç boşluk karakteri koyarakekrana yazdıran bir program tasarlayınız.

Not: Tam sayı bölme işlemlerini ve mod operatörünü kullanın.

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

```
main()
```

```
{
```

```
    int number; /* Kullanıcı tarafından girilecek sayı */
```

```
    int temp1; /* Birinci gecici tamsayı */
```

```
    int temp2; /* İkinci gecici tamsayı */
```

```
    printf( "Bes basamakli bir sayi gir: " );
```

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

```
    printf( "%d ", number / 10000 );
```

```
    temp2 = number % 10000;
```

```
    printf( " %d ", temp2 / 1000 );
```

```
    temp1 = temp2 % 1000;
```

```
    printf( " %d ", temp1 / 100 );
```

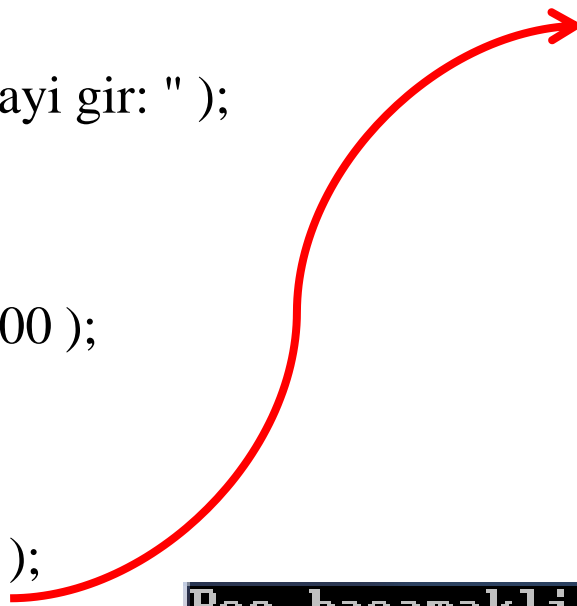
```
    temp2 = temp1 % 100;
```

```
    printf( " %d ", temp2 / 10 );
```

```
    temp1 = temp2 % 10;
```

```
    printf( " %d\n", temp1 );
```

```
}
```



```
Bes basamakli bir sayi gir: 42139
4  2  1  3  9
```

Uygulama 5

69

- Sadece bu bölümde öğrendiğiniz programlama tekniklerini kullanarak 0'dan 10'a kadar olan sayıları karelerini hesaplayıp, sonuçları ekrana aşağıda görüldüğü biçimde yazdıran bir program yazınız.

Sayı	Karesi	Kubu
0	0	0
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000

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

```
main()
```

```
{
```

```
    int count = 0;
```

```
    printf( "\nSayi\tKaresi\tKubu\n" );
```

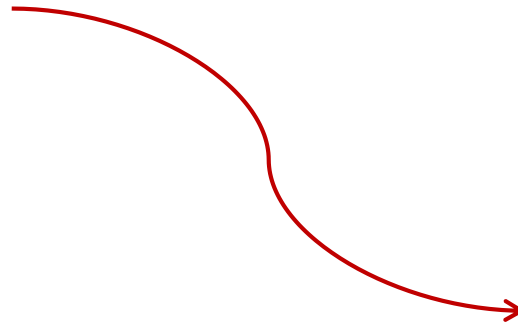
```
    printf( "%d\t%d\t%d\n", count, count * count, count * count * count );
```

```
    count = count + 1;
```

```
    printf( "%d\t%d\t%d\n", count, count * count, count * count * count );
```

```
    count = count + 1;
```

```
    printf( "%d\t%d\t%d\n", count, count * count, count * count * count );
```

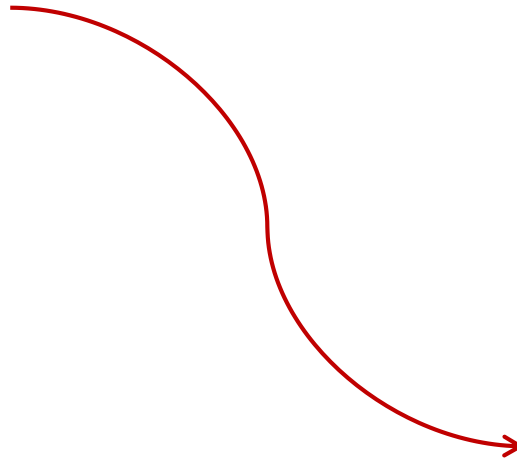


```
count = count + 1;  
printf( "%d\t%d\t%d\n", count, count * count, count * count * count );
```

```
count = count + 1;  
printf( "%d\t%d\t%d\n", count, count * count, count * count * count );
```

```
count = count + 1;  
printf( "%d\t%d\t%d\n", count, count * count, count * count * count );
```

```
count = count + 1;  
printf( "%d\t%d\t%d\n", count, count * count, count * count * count );
```



```
count = count + 1;
printf( "%d\t%d\t%d\n", count, count * count, count * count * count );

count = count + 1;
printf( "%d\t%d\t%d\n", count, count * count, count * count * count );

count = count + 1;
printf( "%d\t%d\t%d\n", count, count * count, count * count * count );

count = count + 1;
printf( "%d\t%d\t%d\n", count, count * count, count * count * count );

}
```


Sorular???

