ALGORİTMA VE PROGRAMLAMA II

HAFTA#12

YZM 1106

Celal Bayar Üniversitesi Hasan Ferdi Turgutlu
Teknoloji Fakültesi

Genel Bakış…

- Bellek ve Adresleme
- Dinamik Bellek Yönetimine Giriş
- Dinamik ve Statik Dizi
- Dinamik Dizi Fonksiyonları
  - Malloc
  - Calloc
  - Realloc
  - Free

7. BÖLÜM

Dinamik Bellek Yönetimi

Bellek ve Adresleme

- Bilgisayarın ana belleği (**RAM**) sıralı kaydetme **gözlerinden** oluşmuştur.
- Her göze bir adres atanmıştır.
- Bu adreslerin değerleri 0 ila belleğin sahip olduğu üst değere bağlı olarak değişebilir.
- Örneğin 1GB bir bellek,
  - 1024*1024*1024 = 1,073,741,824 adet gözden oluşur.

Bellek ve Adresleme (devam…)

- Değişken türlerinin bellekte kapladığı alanlar:

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

int main()
{
  printf( "char
  printf( "short
  printf( "int
  printf( "long
  printf( "unsigned c
  printf( "unsigned s
  printf( "unsigned i
  printf( "unsigned l
  printf( "float
  printf( "double
  printf( "long doubl
  return 0;
}
```

```
char               : 1 bayt
short              : 2 bayt
int                : 4 bayt
long               : 4 bayt
unsigned char      : 1 bayt
unsigned short     : 2 bayt
unsigned int       : 4 bayt
unsigned long      : 4 bayt
float              : 4 bayt
double             : 8 bayt
long double        : 12 bayt
```

Bellek ve Adresleme (devam…)

- Bir programlama dillinde, belli bir tipte değişken tanımlanıp ve bir değer atandığında, o değişkene dört temel özellik eşlik eder:

1. değişkenin adı
2. değişkenin tipi
3. değişkenin sahip olduğu değer (içerik)
4. değişkenin bellekteki adresi

Bellek ve Adresleme (devam…)

- **Örnek:**

```
int tam = 33;
```

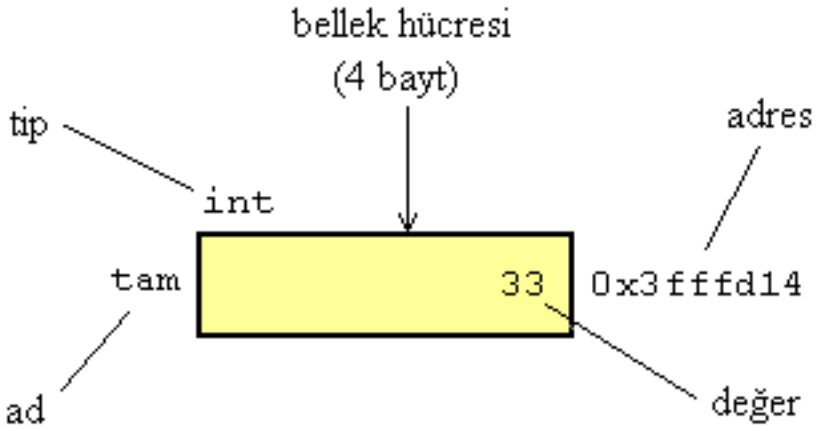
- Bu değişken için, int tipinde bellekte (*genellikle herbiri 1 bayt olan 4 bayt büyüklüğünde*) bir hücre ayrılır ve o hücreye **33 sayısı** ikilik (binary) sayı sitemindeki karşılığı olan 4 baytlık (32 bitlik) karşılığı aşağıdaki gibi yazılır.

00000000 00000000 00000000 00100001

Bellek ve Adresleme (devam…)

- **Örnek:**

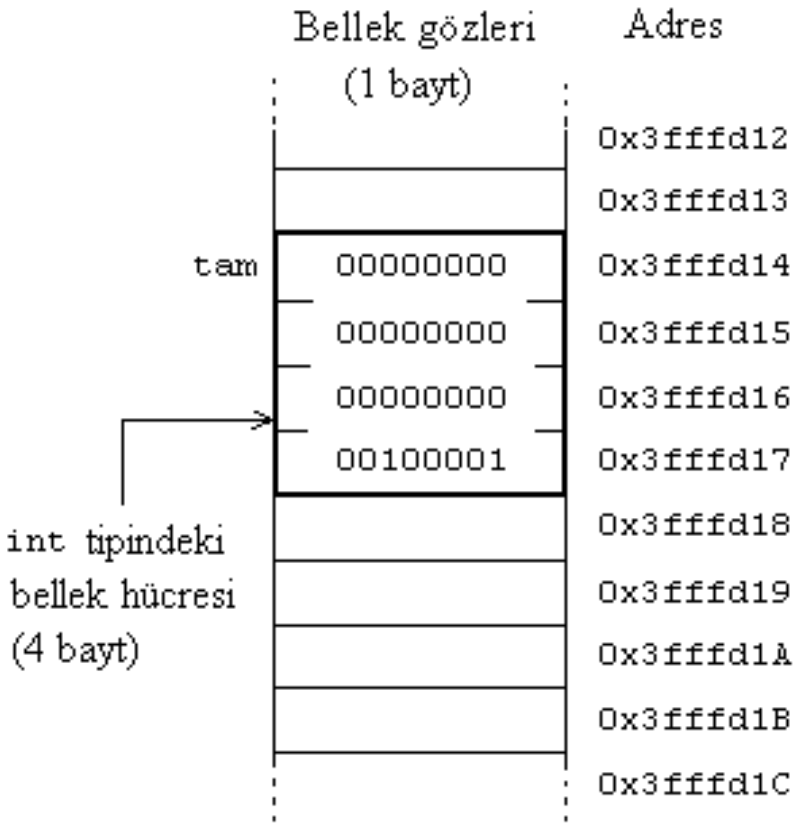
```
int tam = 33;
```



- Bellek adresleri genellikle onaltılık (hexadecimal) sayı sisteminde ifade edilir.
- 0x3fffd14 sayısı onluk (decimal) sayı sisteminde **67108116** sayına karşılık gelir. Bunun anlamı, **tam** değişkeni, program çalıştığı sürece, bellekte **67108116. - 67108120**. numaralı gözler arasındaki **4 baytlık** hücreyi işgal edecek olmasıdır.

Bellek ve Adresleme (devam…)

- **tam** adlı değişkenin bellekteki gerçek konumu ve ikilik düzendeki içeriği aşağıdaki gibidir:



Dinamik Bellek Yönetimi

- Yazılım geliştiricinin dikkat etmesi gereken en önemli noktalardan birisi ürettiği yazılımın sistemin kaynaklarını **en verimli** şekilde kullanması gerekliliğidir.
- Nasıl **sınırsız kullanıcı talebi olamazsa sınırsız sistem kaynağı** da **olamaz**.
- En önemli sistem kaynaklarından birisi de istemci / sunucu tarafında kullanılan BELLEK tir
  - CPU Türü ve Gücü, Disk Türü, Disk Alanı Boyutu, Toplam Sunucu Sayısı, network bandwidth vb.

Dinamik Bellek Yönetimi (devam…)

- Eğer yazılımcı bellek yönetimini iyi yapamıyorsa doğal olarak **daha fazla belleğe** ihtiyaç duyar.
- Daha fazla belleğe ihtiyaç duymak belli başlı birkaç sorunu beraberinde getirir.
  - Fazla maliyet,
  - Programın yavaş çalışması

  bunların en önemlileridir.

Statik ve Dinamik Dizi

Statik Dizi

- Bir C programı içerisinde, dizilerin boyutu ve kaç elemanlı olduğu *program başında belirtilirse*, derleyici o dizi için gereken bellek alanını (bölgesini) program sonlanıncaya kadar **saklı tutar** ve bu alan başka bir amaç için kullanılamaz.
- Bu türdeki diziler **statik dizi** olarak adlandırılırlar.
- Statik dizinin boyutu programın çalışması esnasında (run time) değiştirilemez.

Statik ve Dinamik Dizi (devam…)

Dinamik Dizi

- **Programın çalışırken** bir dizinin boyutu ve eleman sayısı bazı yöntemler kullanılarak değiştirilebilir.
- Bu tür dizilere **dinamik dizi** denir.
- Dinamik diziler için gereken bellek bölgesi, derleyici tarafından
  - işletim sisteminden **istenir**,
  - **kullanılır** ve
  - daha sonra istenirse bu bölge **boşaltılır**.

Statik ve Dinamik Dizi (devam…)

Örneğin:

- `char ad[20];`

- İfadesinde derleyici, bellekten 20 byte boyutunda sürekli bir alan tahsis edecektir.
- Bu yer tahsisi, program başlatılmadan önce yapılmaktadır.
- Yani program çalışırken bu dizinin boyutunu değiştirmeniz mümkün değildir.
- Fakat bazı durumlarda (**genelde** ☺) bellekten boyutu sabit olmayan ve sürekli değişebilen yerler tahsis etmemiz gerekecektir.

Statik ve Dinamik Dizi (devam…)

Örneğin:

- Telefon rehberi yazılımı geliştirmek için bellekten *tahsis edeceğiniz bölgenin boyutunu önceden tahmin edebilir misiniz*?
- Her yeni telefon eklendiğinde bellekte ayırdığınız yeri büyüterek belleği en verimli şekilde kullanmak durumundasınız.
- *Yani dinamik bellek yönetimi ile programın çalışma zamanı sırasında (Run-Time)-işletim sistemine danışarak- sürekli bellek bölgeleri tahsis ederiz.*

Dinamik Dizi Fonksiyonları

Dinamik Bellek Fonksiyonu	Açıklama
void *malloc(**size_t** eleman_sayısı);	Bellekte her biri **size_t** tipinde olan **eleman_sayısı** kadar yer (bellek bloğu) ayırır. Bu yer verilmezse NULL gönderir.
void *calloc(**size_t** eleman_sayısı, size_t **nbayt**);	Bellekte her biri **nbayt** kadar yer işgal edecek **eleman_sayısı** kadar boş yer ayırır ve bütün bitleri sıfırlar. Bu yer ayrılamazsa geriye NULL gönderir.
void *realloc(void ***ptr**, size_t **nbayt**);	**ptr** işaretçisi ile gösterilen bellek bloğunu, **nbayt** kadar büyüterek veya küçülterek değiştirir. Bu iş gerçekleşmezse geriye NULL gönderir.
void free(void ***ptr**);	Daha önce ayrılan adresi **ptr'de** saklanan bellek alanının boşaltır.

Statik ve Dinamik Dizi (devam…)

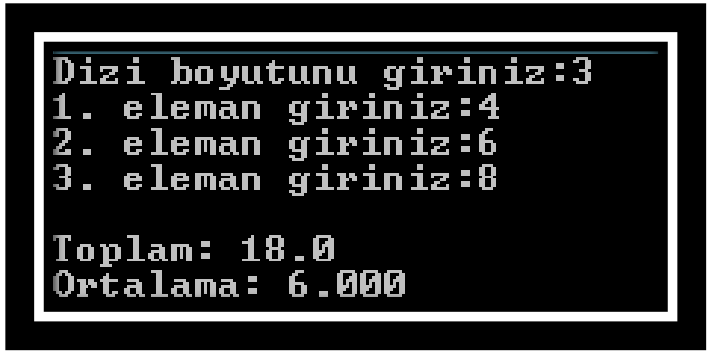

```
int *dizi; /* dinamik dizi bildirimi */
scanf("%d",&n); /* eleman saysını belirle */

/* n tane bellek bloğu isteniyor */
dizi = (int *) malloc( sizeof(int)*n );

/* Boş yer varmı sorgulanıyor */
if( dizi == NULL )
  printf("Yetersiz bellek alanı\n"), exit(1);
     ...
/* dizi burada kullanılıyor */
/* bellek bloğu boşaltılıyor */
free(dizi);
```

Örnek 1: Dinamik Dizi Ortalama Hesabı

- Sayısal türde boyutu dinamik olarak dışarıdan girilecek bir dizi tanımlanacaktır (int *dizi).
- Dizinin boyutu (n değişkeni) klavyeden kullanıcıya girdirilmeli ve dinamik dizi malloc fonksiyonu kullanılarak oluşturulmalıdır.
- Dizi boyutu girildikten sonra dizinin her bir için değerler klavyeden girilmelidir.
- Son olarak dizi elemanlarının toplamı ve ağırlıklı ortalamaları yazdırılacaktır.



Örnek 1: Dinamik Dizi Ortalama Hesabı


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

int main()
{
    int *dizi, i;
    int n; //dizi boyutu
    float toplam = 0;

    printf("Dizi boyutunu giriniz:");
    scanf("%d", &n);

    dizi = (int *) malloc(sizeof(int)*n);

    for (i=0; i<n; i++)
    {
      printf("%d. eleman giriniz:",i+1);
      scanf("%d",&dizi[i]);
    }

    for (i=0; i<n; i++)
      toplam += dizi[i];

    printf("\nToplam: %.1f\n",toplam);
    printf("Ortalama: %.3f\n",(toplam/n));

    free(dizi);
    return 0;
}
```

Örnek 1: Dinamik Dizi Ortalama Hesabı

EKLENTİ

- Uygulamanın bir döngü içerisinde çalışması sağlanacaktır.
- Dizi boyutu olan n değeri 0 girilmediği sürece uygulama çalışmaya devam etmelidir.

malloc() ve calloc() Farkı

- malloc() fonksiyonu tahsis ettiği bellekteki bölgelere herhangi bir ilk değer atama işlemi uygulamaz.
- Yani bellekteki değerleri ile beraber size tahsis eder, ilk değer atama işlemi yazılımcıya kalmıştır.
- calloc() fonksiyonu bellek tahsisatı yaparken malloc() fonksiyonunu kullanır.
- Farklı olarak ayırdığı bellek bölgesini sıfırlamaktadır.

Örnek 2: malloc() ve calloc() Farkı


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

int main()
{
    int *p, i;

    //p = calloc(15, sizeof(int));

    p = (int *) malloc(sizeof(int)*15);

    for(i = 0; i < 15; i++)
        printf("%d. eleman ---> %d\n", i + 1, p[i]);

    free(p);

    return 0;
}
```

Örnek 3: realloc()

- `malloc()` fonksiyonunu kullanarak **10 karakter kapasiteli** bir işaretçi tanımlayınız.
- Bu işaretçiye `strcpy()` fonksiyonunu kullanarak **"Algoritma"** değerini atayınız.
- Bu işaretçinin kapasitesini `realloc()` fonksiyonunu kullanarak **25 karaktere çıkartın**.
- İşaretçinin sonuna `strcat()` fonksyionunu kullanarak **"veProgramlama"** ekleyin ve ekrana yazdırın.

Örnek 3: realloc()


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

int main()
{
   char *str;

   /* Initial memory allocation */
   str = (char *) malloc(10);
   strcpy(str, "Algoritma");
   printf("Deger = %s,  Adres = %u\n", str, str);

   /* Reallocating memory */
   str = (char *) realloc(str, 25);
   strcat(str, "veProgramlama");
   printf("Deger = %s,  Adres = %u\n", str, str);

   free(str);

   return 0;
}
```

Örnek 4: Karakter Dizisinin Alt Parçalarını Bulma (substring)

- Girilen bir karakter dizisinin istenilen pozisyondan itibaren istenilen karakter sayısı kadar kopyalanmasını sağlayan C programını yazınız.

1. 100 elemanlı dizi isimli bir karakter dizisi tanımlanacaktır.
2. İlk olarak klavyeden bir karakter dizisi girilecektir. Burada gets() fonksiyonu kullanılarak dizi doldurulacaktır.
3. Daha sonra dizinin hangi karakterden itibaren kopyalanacağı bilgisi okunarak, sayısal pozisyon değişkenine atanacaktır.
4. Daha sonra kopyalanacak karakter sayısı girilecek ve sayısal uzunluk değişkenine atanacaktır.

Örnek 4: Karakter Dizisinin Alt Parçalarını Bulma (substring)


```
Karakter dizisini giriniz: Algoritma ve Programlama
Kopyalama baslangic pozisyonunu giriniz: 10
Kopyalanacak karakter sayisini giriniz: 13
Sonuc: " ve Programla"
```

5. Prototipi aşağıdaki gibi olan bir fonksiyon yazılarak kopyalanan değer ekranda gösterilecektir.

char *SubString(**char** *dizikopya, **int** pozisyon, **int** uzunluk)

KAYNAKLAR

- N. Ercil Çağıltay ve ark., C DERSİ PROGRAMLAMAYA GİRİŞ, Ada Matbaacılık, ANKARA; 2009.
- Milli Eğitim Bakanlığı "Programlamaya Giriş ve Algoritmalar Ders Notları", 2007
- C Programlama Dili, Şerafettin ARIKAN
- Problem Solving and Program Design in C, Hanly, Koffman
- http://www.AlgoritmaveProgramlama.com

Algoritma ve Programlama

İYİ ÇALIŞMALAR…