Algoritma ve Programlama -1

12. HAFTA

Dinamik Bellek Yönetimi ve Yapılar

11. Hafta - Tekrar

- Göstericiler ve Diziler
- Göstericiler ve Fonksiyonlar
- Karakter Göstericileri

Dinamik Bellek Yönetimi

- Diziler derleyiciler tarafından, derleme aşamasında ele alınır.
  - Yani programın çalışma zamanı sırasında bir dizinin uzunluğu değiştirilmez.

- Birçok uygulamada dizilerin ne kadar uzunlukta açılacağı programın çalışma zamanı sırasında ve birtakım işlemlerden sonra belirlenebilmektedir.
  - Dersi geçecek öğrencilerin sayısı
  - Kombine alacak taraftar sayısı

- Bu gibi durumlar nedeniyle diziler için ‘en kötü olasılığı gözönünde bulundurarak’ boyut belirlemek zorunda kalınmaktadır.

- Bu yöntem ise belleğin verimsiz kullanılmasına sebep olur.
  - Açılan diziler yerel ise ilgili blok sonlanana, global ise sürekli olarak bellekte tutulacaktır.

- Programın çalışma zamanı sırasında belli sayıda sürekli bellek bölgesinin tahsis edilmesine ve istenildiğinde geri bırakılmasına olanak sağlayan yöntemlerin kullanılmasına ***dinamik bellek yönetimi*** denir.

Standart Dinamik Bellek Fonksiyonları

- Tüm sistemlerde aynı isimle ve aynı işlevleri yerine getirecek biçimde bulunan fonksiyonlardır.

- Bunlar;
  - malloc
  - calloc
  - realloc

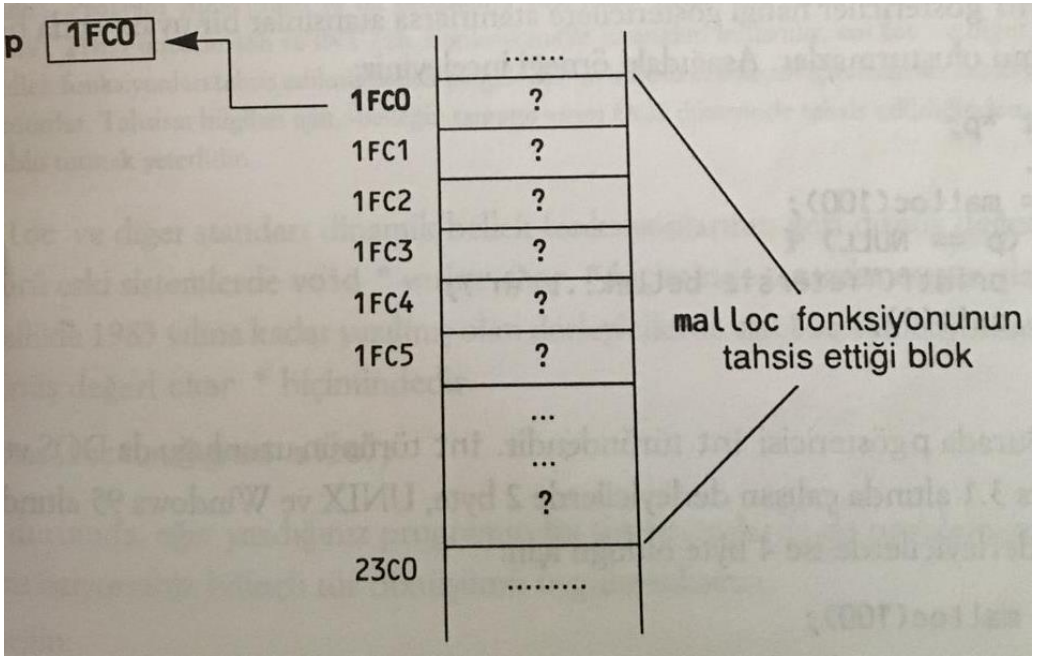
malloc Fonksiyonu

- En çok kullanılan dinamik bellek fonksiyonudur.
  - void* malloc (unsigned size);
- Programın çalışma zamanı sırasında sisteme danışarak belleğin güvenli bir bölgesinde parametresi ile belirtilen byte kadar uzunlukta sürekli bellek bölgesi tahsis eder.
- Geri dönüş değeri olarak tahsis ettiği sürekli bloğun başlangıç adresini vermektedir.
- Herhangi bir nedenden ötürü tahsisat işlemi yapamazsa 0 değeri (NULL pointer) geri döner.
  - Tahsisatın başarılı bir şekilde yapılıp yapılmadığı mutlaka kontrol edilmelidir.

malloc Fonksiyonu

- Burada malloc fonksiyonundan sisteme danışarak 10 byte sürekli bellek tahsis etmesi istenmiştir.
- Fonksiyonun geri dönüş değeri p göstericisine atanmıştır.
- malloc ile tahsis edilen bloğun içerisinde rastgele değerle bulunmaktadır. Herhangi bir ilk değer verme işlemi uygulanmaz.

```
char *p;
...

p = malloc (1024);
if (p == NULL) {
    printf("Yetersiz bellek!..\n");
    exit(1);
}
...
```



malloc Fonksiyonu

- malloc fonksiyonun geri dönüş değeri daha önce belirtildiği gibi void'dir.
- void göstericiler hangi türdeki göstericilere atanırsa atansın bir uyarı ya da hata durumu oluşturmazlar.
  - p=malloc(sizeof(int) * 50)
- Yukarıda belirtildiği gibi bir tanımlama uyarı ya da hata mesajı vermese de bu tip durumlarda bilinçli tür dönüşümü yapmak çok daha kullanışlıdır.
  - Eski tip derleyicilerde bu tip durumda hata alınabilir.
  - p= (int *) malloc(sizeof(int) *50);
- Yandaki örnek programda, malloc ile tahsis edilecek toplam alan (heap) bulunmaktadır.

```
void main(void)
{
    long t = 0;
    char *p;

    for (;;) {
        p = malloc(1024);
        if (p == NULL)
            break;
        t += 1024;
    }
    printf("Toplam alan: %ld\n", t);
}
```

calloc Fonksiyonu

- İşlevsel olarak malloc fonksiyonuna benzer.
- void *calloc (unsigned count, unsigned size)
  - calloc birinci parametresiyle belirtilen sayı ile ikinci parametresinde belirtilen sayının çarpımı kadar sürekli belleği tahsis etmek amacıyla kullanılır.
  - int *p = (int *) calloc (50, sizeof(int));
- calloc tahsisat yapabilmek amacıyla içerisinde malloc fonksiyonunu kullanır.
- Fakat calloc, malloc fonksiyonundan farklı olarak tahsis ettiği bellek bloğunu sıfırlar.

malloc vs calloc

- The name **malloc** and calloc() are library functions that allocate memory dynamically.
  - **Initialization:**
    - malloc() allocates memory block of given size (in bytes) and returns a pointer to the beginning of the block.
    - malloc() doesn't initialize the allocated memory. If we try to acess the content of memory block then we'll get garbage values.
    - calloc() allocates the memory and also initializes the allocates memory block to zero. If we try to access the content of these blocks then we'll get 0.
  - **Number of arguments:** Unlike malloc(), calloc() takes two arguments:
    - Number of blocks to be allocated.
    - Size of each block.

```
int *p;
int k;
...
p = (int *) malloc(sizeof(int) * 100);
if (p == NULL) {
    printf("Yetersiz bellek!...\n");
    exit(1);
}
for (k = 0; k < 100; ++k)
    p[k] = 0;
...
```

```
int *p;
...
p = (int *) calloc(100, sizeof(int));
if (p == NULL) {
    printf("Yetersiz bellek!...\n");
    exit(1);
}
...
```

realloc Fonksiyonu

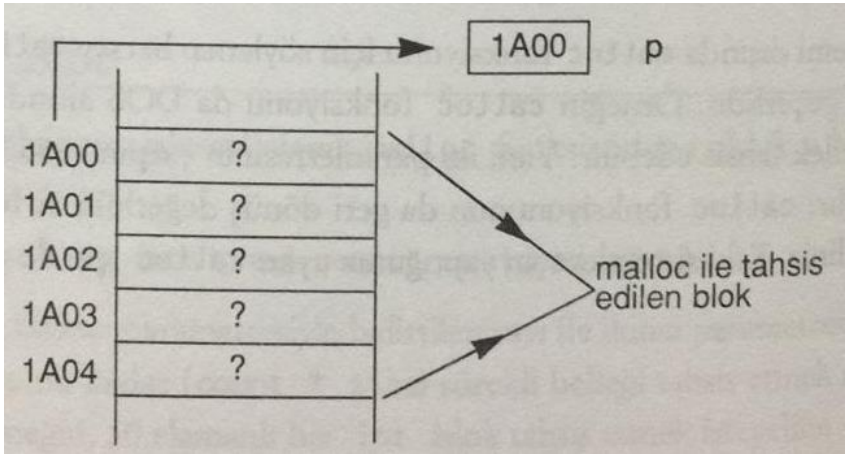
- Daha önce malloc ya da calloc fonksiyonlarıyla tahsis edilmiş bellek bloğunu büyütmek ya da küçültmek için kullanılır.

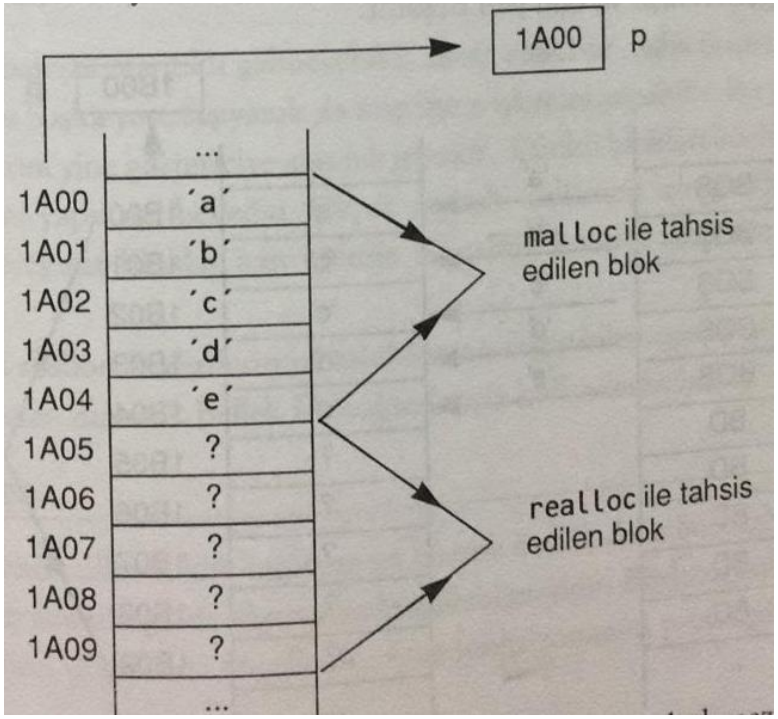
- void *realloc(void *block,unsigned newsize);
  - Birinci parametre, daha önce tahsis edilen bellek bloğunun başlangıç adresini, ikinci parametresi ise bloğun toplam yeni uzunluğudur.

- realloc, daha önce tahsis edilen bloğun hemen altında sürekliliği bozmayacak biçimde tahsisat yapar.
  - Eğer istenilen uzunlukta yer bulamazsa, bu sefer bloğun tamamı için bellekte başka yer araştırır.
  - Yer bulursa bulduğu yeri tahsis eder ve eski değerleri buraya taşır.
  - Bulamazsa NULL değeriyle geriye döner.

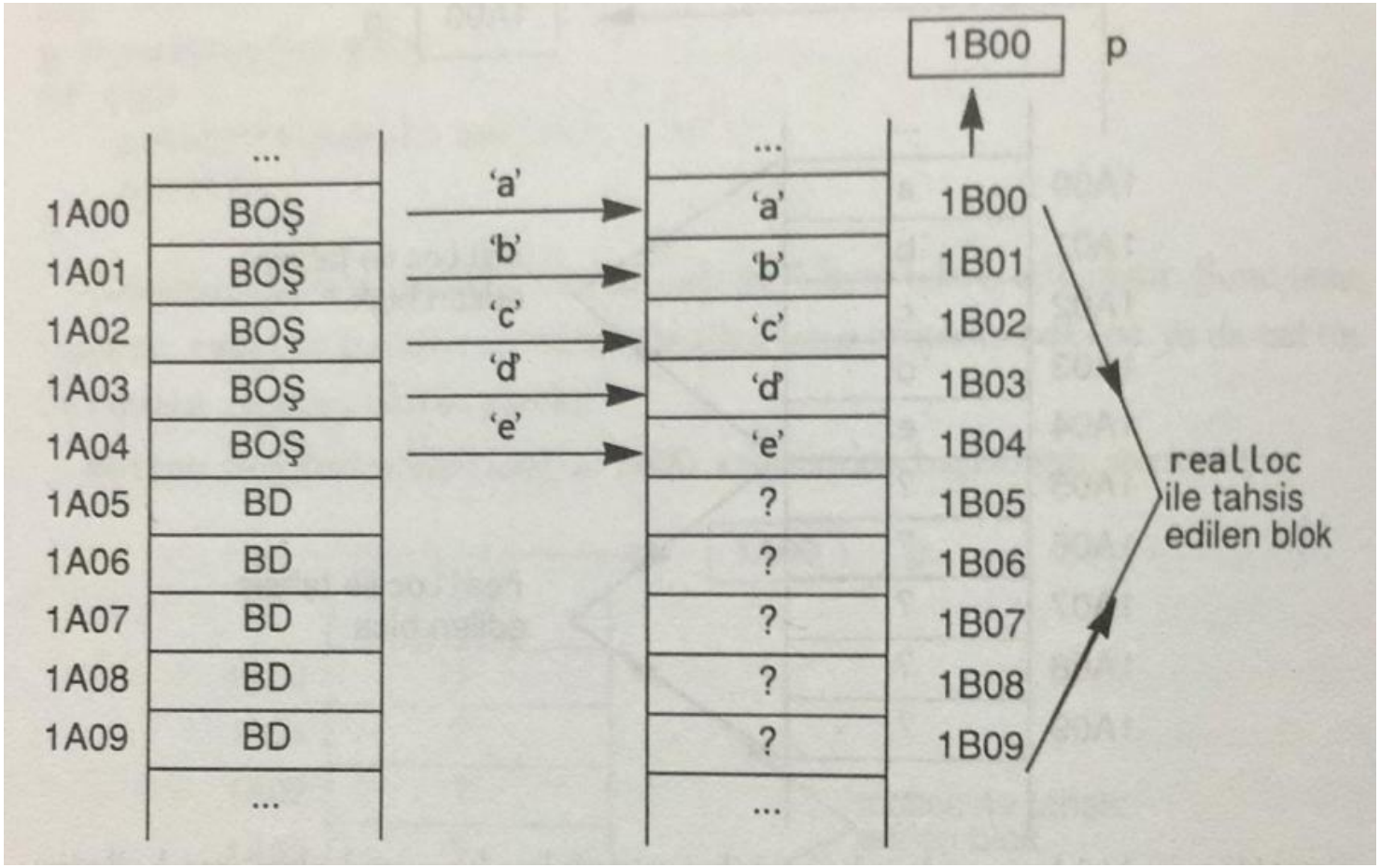
realloc

```
char *p;
...
p = malloc(5);
if (!p) {
    printf("Yetersiz bellek!...\n");
    exit(1),
}
for (k = 0; k < 5; ++k)
    p[k] = 'a' + k;
...
p = realloc(p, 10);
if (!p) {
    printf("Yetersiz bellek!...\n");
    exit(1),
}
```





realloc Fonksiyonu



realloc Fonksiyonu

- realloc fonksiyonunun tahsis edilmiş bloğu başka bir bölgeye taşıma durumu olması nedeniyle göstericiye tekrar atanmalıdır.

- realloc fonksiyonun tahsis ettiği blok içerisinde rastgele değerler vardır; tahsis ettiği bloğa ilkdeğer vermez.

- Sadece bellek bloğunu büyütmek için değil aynı zamanda küçültmek amacıyla da kullanılabilir.

```
char *p;
...
p = malloc(100);
if (p == NULL) {
    printf("Yetersiz bellek!...\n");
    exit(1);
}
...
p = realloc(p, 50);
```

- İlk bakışta mantıklı gelmese de, realloc daha önce tahsis edilmiş olan bloğu başka yere taşıyarak da küçültme işlemi yapabilir.

free Fonksiyonu

- Bellek yönetime dinamiklik sağlayan en önemlik özellik daha önce tahsis edilmiş olan blokların istenildiğinde sisteme iade edilebilmesidir.
- Böylece kullanılmayan blokların verimi düşürmesi engellenir.
- void free (void *block);
- Bu fonksiyon daha önce tahsis edilmiş bellek bloğunu geri boşaltarak sisteme iade eder.
- Tahsis edilmiş bloklar free fonksiyonu ile boşaltılmamışsa programın sonlanmasıyla otomatik olarak boşaltılırlar.

Dynamic Memory Allocation

- An array is a collection of fixed number of values of a single type. That is, you need to declare the size of an array before you can use it.

- Sometimes, the size of array you declared may be insufficient. To solve this issue, you can allocate memory manually during run-time. This is known as dynamic memory allocation in C programming.

Syntax of malloc()

```
ptr = (cast-type*) malloc(byte-size)
```

Example:

```
ptr = (int*) malloc(100 * sizeof(int));
```

Syntax of calloc()

```
ptr = (cast-type*)calloc(n, element-size);
```

Example:

```
ptr = (float*) calloc(25, sizeof(float));
```

Syntax of free()

```
free(ptr);
```

Syntax of realloc()

```
ptr = realloc(ptr, x);
```

malloc,calloc and free

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

int main()
{
     int n, i, *ptr, sum = 0;

     printf("Enter number of elements: ");
     scanf("%d", &n);

     ptr = (int*) malloc(n * sizeof(int));
     if(ptr == NULL)
     {
          printf("Error! memory not allocated.");
          exit(0);
     }

     printf("Enter elements: ");
     for(i = 0; i < n; ++i)
     {
          scanf("%d", ptr + i);
          sum += *(ptr + i);
     }

     printf("Sum = %d", sum);
     free(ptr);
     return 0;
}
```

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

int main()
{
     int n, i, *ptr, sum = 0;
     printf("Enter number of elements: ");
     scanf("%d", &n);

     ptr = (int*) calloc(n, sizeof(int));
     if(ptr == NULL)
     {
          printf("Error! memory not allocated.");
          exit(0);
     }

     printf("Enter elements: ");
     for(i = 0; i < n; ++i)
     {
          scanf("%d", ptr + i);
          sum += *(ptr + i);
     }

     printf("Sum = %d", sum);
     free(ptr);
     return 0;
}
```

realloc

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

int main()
{
     int *ptr, i , n1, n2;
     printf("Enter size of array: ");
     scanf("%d", &n1);

     ptr = (int*) malloc(n1 * sizeof(int));

     printf("Addresses of previously allocated memory: ");
     for(i = 0; i < n1; ++i)
            printf("%u\n",ptr + i);

     printf("\nEnter new size of array: ");
     scanf("%d", &n2);
     ptr = realloc(ptr, n2 * sizeof(int));

     printf("Addresses of newly allocated memory: ");
     for(i = 0; i < n2; ++i)
            printf("%u\n", ptr + i);
     return 0;
}
```

Uygulamalar

- Dinamik bellek ile integer dizi kullanımı
- Dinamik bellek ile string kullanımı
- Bir stringin karakterlerini tek tek ekrana yazdırma

Yapılar

- Struct birden fazla alt alandan oluşan veri tipidir.
- Aralarından mantıksal bir ilişki bulunan farklı türden bilgiler, Struct kullanılarak mantıksal bir bütün olarak ifade edilebilirler.
- Diziler gibi belleğe sürekli bir biçimde yerleşirler.
- Dizilerde olduğu gibi başlangıç adresi geçirilerek fonksiyonlara kolaylıkla aktarılabilirler.

Yapıların Bildirimi

```
struct [yapı_ismi] {
        <veri tipi> <yapı_elemanı>;
        <veri tipi> <yapı_elemanı>;
        <veri tipi> <yapı_elemanı>;
        …
        } ;
```

```
struct bilgi{
        int id;
        char ad[15];
        char soyad[15];
        } ;
```

```
struct tarih{
        int day;
        int month;
        int year;
        } ;
```

- Yapı bildirimleriyle derleyici yalnızca yapılar hakkında bilgi edinir; bellekte onlar için herhangi bir yer ayırmaz.
- Yapı bildirimlerini bir şablon gibi düşünebilirsiniz.
- Tıpkı fonksiyon prototiplerinde olduğu gibi derleyici bilgilendirmek amacıyla kullanılır.

Yapı Değişkenlerinin Tanımlanması

- struct <yapı_ismi> <yapı_değişkenin_ismi>;

```
struct nokta{
        int x;
        int y;
        } ;
 struct nokta a;
```

```
struct bilgi{
        int id;
        char ad[15];
        char soyad[15];
        } ;
 struct bilgi test;
```

- sizeof(a) ve sizeof(b) kaçtır?

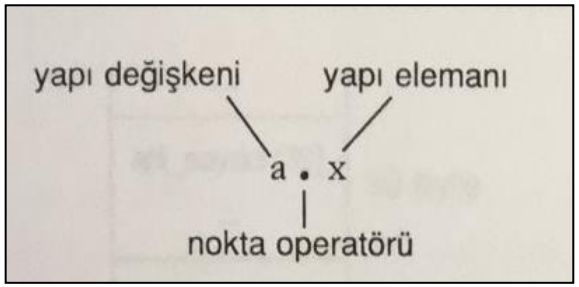
- Yapı bildirimleri ile tanımlama işleminin birlikte yapılması

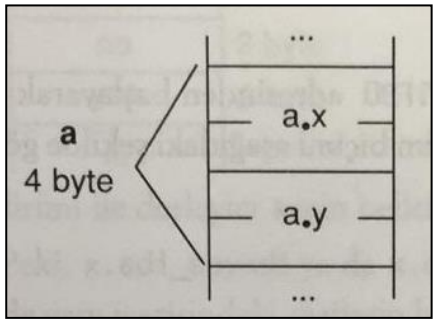
```
struct nokta{
        int x;
        int y;
        } a, b, c ;
```

Yapı Elemanlarına Erişme ve Nokta Operatörü

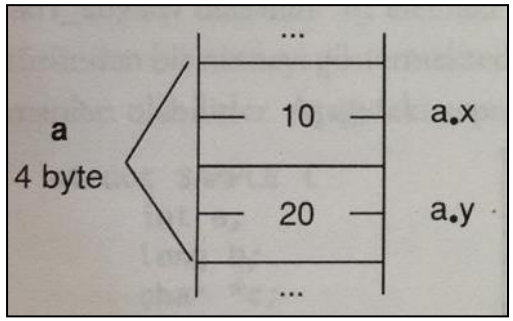
- Bir yapı değişkeni gerçekte kendi içerisinde yapı elemanlarını barındıran bileşik bir değişkendir.
- Yapı elemanlarını nokta operatörü ile erişilir.

```
struct nokta {
        int x;
        int y;
        };
 struct nokta a;
```





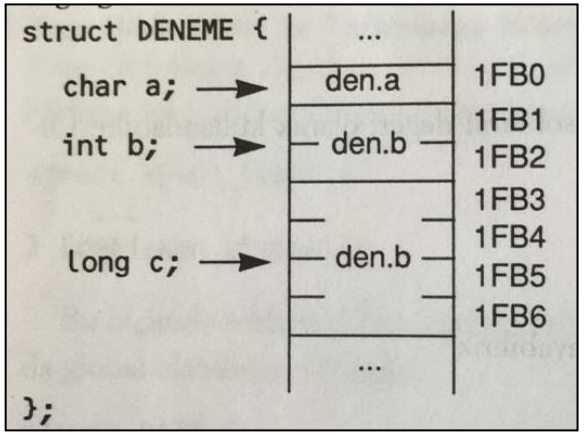
a.x=10;
a.y=20;



Yapı Elemanlarının Bellekteki Yerleşimi

- Bir yapı değişkenin elemanları tıpkı dizilerde olduğu gibi belleğe sürekli bir biçimde yerleşir.
- Yapı elemanlarının yerleşimi oldukça basit bir kural ile belirlenmiştir.
- Yapı bildirimde ilk belirtilen eleman belleğin düşük anlamlı adresinde bulunacak biçimde yerleştirilir.

```
struct DENEME {
    char a;
    int b;
    long c;
};
...
struct DENEME den;
```



Nesne	Tür	Uzunluk
den	struct DENEME	7
den.a	char	1
den.b	int	2
den.c	long	4

Uygulamalar

- Aşağıda verilen öğrenci bilgilerinin bir yapıda tutulmasını sağlayınız.
  - Ad, Soyad, Numara, Sınıf

- Kullanıcıdan alınan doğum tarihi ve mevcut gün bilgisi ile kullanıcının yaşını hesaplayan programı struct kullanarak gerçekleştiriniz.

Yapı Elemanları Olarak Diziler ve Göstericiler

- Diziler ve göstericiler de yapı elemanı olabilirler.

```
struct INSAN{
        char adi_soyadi[30];
        char dogum_yeri[30];
        int no;
        int yas;
        } ;
 struct INSAN x;
```

```
struct SAMPLE{
        int a;
        long  b;
        char *c;
        } ;
 struct SAMPLE smp;
```

x.adi_soyadi[n] nedir?

smp.c'ye doğrudan atama yapılabilir mi?

Uygulama

Yapı Bildirimlerinin Yapılış Yerleri

```
void main()
{
    struct DATE {
        int day, month, year;
    };
    struct DATE d;
    ...
}
fonk()
{
    struct DATE x;
    ...
}
```

yapı yerel olarak bildirilmiş

Değişken tanımlaması yapılabilir

Hata! yapı yerel olarak bildirildiği için burada değişken tanımlaması yapılamaz

```
struct DATE {
    int day, month, year;
};

void main()
{
    struct DATE d;
    ...
}

fonk()
{
    struct DATE x;
    ...
}
```

Yapı global olarak bildirilmiş

Değişken bildirimi yapılabilir

DATE yapısı global olarak bildirildiği için değişken bildirimi yapılabilir

Yapı Değişkenlerine İlkdeğer Verilmesi

```
struct SAMPLE{
        char *p;
        int n;
        float f;
        } ;
```

struct SAMPLE x={"örnek", 100, 5.78} ;

struct DATE date={1, 4, 1995} ;

```
struct DATE{
        int day,month,year;
        } ;
```

```
struct NO{
        int x[5];
        char *p;
        } ;
```

struct NO n={ {10,20,30},"Stokta var"} ;

Yapı Göstericileri

- Yapı değişkenlerinin de adresleri bulunmaktadır.

```
struct DENEME{
        char ch;
        int n;
        long l;
        } ;
 struct DENEME x;
```

- &x.ch, &x.n ve &x.l ile yapı elemanlarının adresleri alınabildiği gibi x yapısının adresi de &x ile alınabilir.
- &x ile elde edilen adresi struct DENEME türündendir.
- Her türden adresi içine alınabilecek bir gösterici tanımlandığına göre yapılar için de göstericiler tanımlanabilir.
  - struct DENEME *p;
- (*p).ch='a'; ve *p.ch='a' ???
- -> OK (arrow) operatörü, p->ch='a'

Uygulama

İç İçe Yapılar

- Bir yapının içinde başka bir yapı tanımlanabilir.
- İç içe yapıların tanımlanması C'de iki biçimde yapılmaktadır.

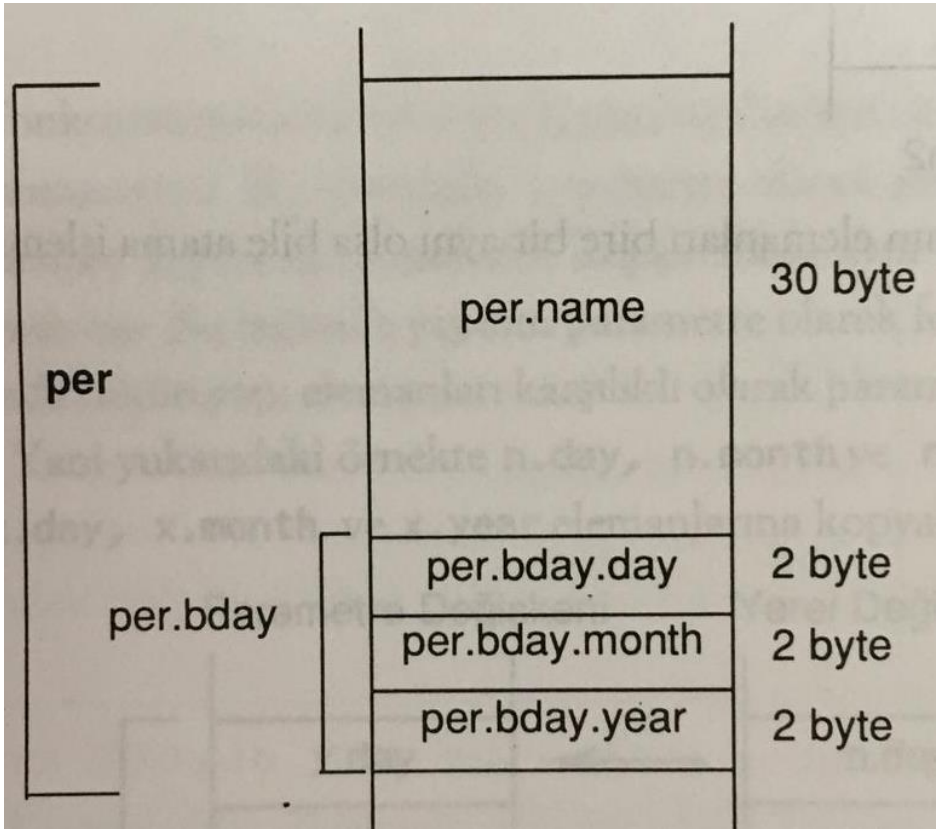
```
struct DATE{
        int day,month,year;
 };

struct PERSON{
        char name[30];
        struct DATE bday;
 };
```

```
struct PERSON {
        char name[30];
        struct DATE{
                int day,month,year;
         } bday;
 };
```

Uygulama

İç İçe Yapılar

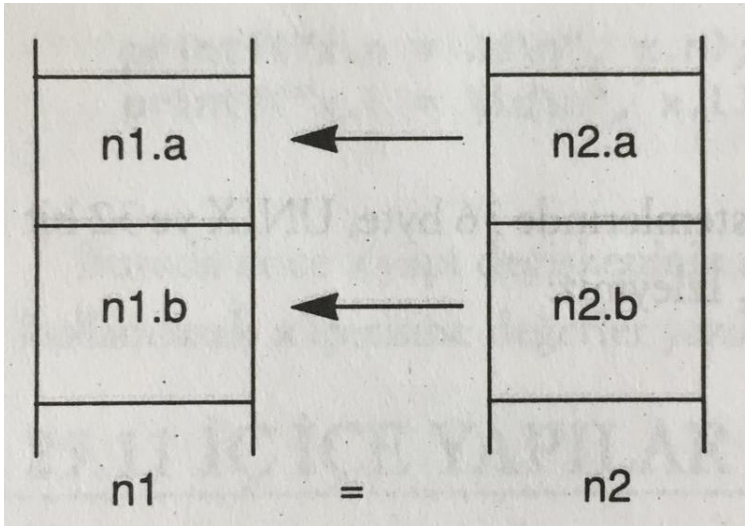
```
struct PERSON per;
```



Yapı Değişkenleri Arasındaki İşlemler

- C'de yalnızca aynı türden iki yapı değişkeni birbirlerine atanabilir.
- Bunun dışında yapı değişkenleri toplama,çıkarma gibi aritmetik ya da mantıksal işlemlere sokulamazlar.
- Örneğin; aşağıdaki durumda n2 yapı değişkeninin elemanları n1 yapı değişkenin elemanlarına atanır.

```
struct X {
        int a;
        double b;
};
struct X n1,n2;
...
n1=n2;
```



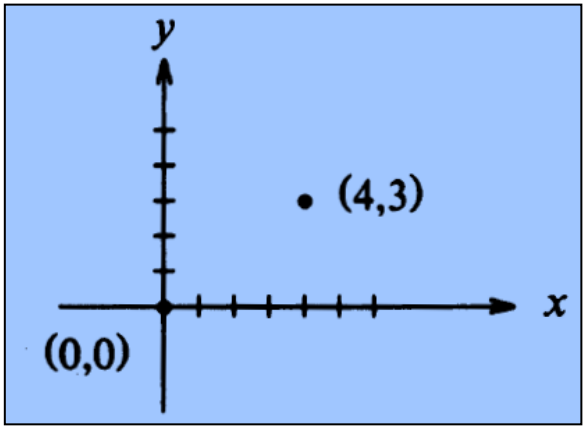
Yapı Değişkenleri Arasındaki İşlemler

- İsimleri farklı olan iki yapının elemanları bire bir aynı olsa bile atama işlemi geçersizdir.

```
struct X1{
        int a;
        double b;
 } ;

struct X2{
        int a;
        double b;
 } ;
```

n1=n2;

işlemi geçersizdir!

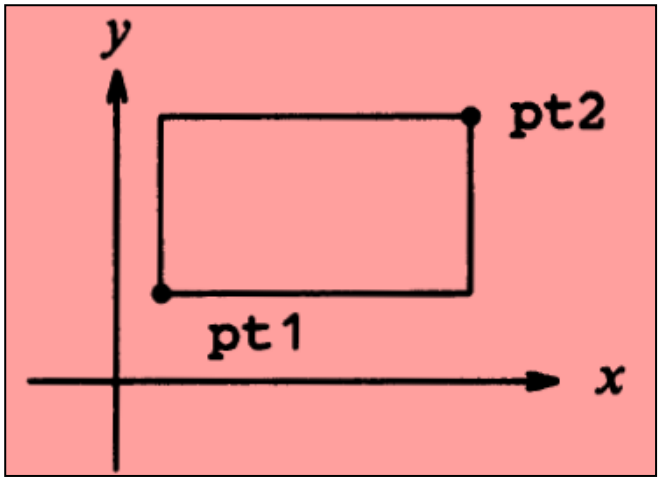



```
struct point {
    int x;
    int y;
};
```

```
struct point pt;
```

```
struct point maxpt = { 320, 200 };
```

```
printf("%d,%d", pt.x, pt.y);
```




```
struct rect {
    struct point pt1;
    struct point pt2;
};
```

```
struct rect screen;
```

```
screen.pt1.x
```

Yapılar ve Fonksiyonlar

```
struct point {
    int x;
    int y;
};
```

```
struct rect {
    struct point pt1;
    struct point pt2;
};
```

```
/* makepoint:  make a point from x and y components */
struct point makepoint(int x, int y)
{
    struct point temp;

    temp.x = x;
    temp.y = y;
    return temp;
}
```

```
struct rect screen;
struct point middle;
struct point makepoint(int, int);

screen.pt1 = makepoint(0, 0);
screen.pt2 = makepoint(XMAX, YMAX);
middle = makepoint((screen.pt1.x + screen.pt2.x)/2,
                   (screen.pt1.y + screen.pt2.y)/2);
```

Uygulama

Yapılar ve Diziler

- Diziler aynı türden elemanları tutmaktadır.

- Yani her bir elemanı bir yapı değişkeni olan bir dizi tanımlanabilir.

```
struct point {
      int x;
      int y;
 } ;
 struct point noktalar[20];
```

Problem – Maaş Hesabı

- Bir şirket tüm personelleri için maaş zamlarının hesaplanmasını istemektedir.
- Şirket maaş zammı hesaplaması esnasında personel id'si, ad soyadı, mevcut maaşı, çocuk sayısı ve önceki zam miktarını dikkate almaktadır.
- Şirkette mevcut durumda 10 personel çalışmaktadır. Personel sayısı zamanla değişmektedir.
  - Şirket en fazla 100 personel bünyesinde bulundurabilmektedir.
- Zam oranı;
  - Maaşı 2000'den küçük olanlar için %20,
  - 2000 ve 3000 arasında olanlar için %15,
  - 3000' den büyük olanlar için %10

  Belirlenmektedir.

- Ayrıca şirket, personele her bir çocuk için 30 TL ekstra zam vermektedir.
- Şirket, personele yapılan yeni zam miktarı eski zam miktarından daha az olması durumda eski maaş zammını dikkate almaktadır.
- Şirkette bulunan tüm personeller için bir yapı oluşturan ve hepsinin maaşlarının hesaplanmasını sağlayan programı yazınız.

Önümüzdeki Hafta

- Yapılar (Struct) Uygulama
- Dosya İşlemleri
- Dosyadan Veri Okuma / Yazma

Kaynakça

1. Aslan, K., A'dan Z'ye C Kılavuzu, Pusula Yayınları.
2. https://www.geeksforgeeks.org/calloc-versus-malloc/
3. https://www.programiz.com/c-programming/c-dynamic-memory-allocation
4. Kernighan, K.W., Ritchie, D.M., The C Programming Language, Prentice Hall Software Series.