Dynamic – Static Memory Allocation

By Mustafa Onur Parlak

- **Dynamic Memory Allocation**
  - **Malloc, realloc, calloc** ile yapılır.
  - Kendi kendine yok edilmemesi bir problemdir. Bu sebeple yazılımı yazarken dikkatli olunması gerekmektedir.
  - Fonksiyonlar ayrıldıkları memory'nin **başlangıç adresini** dönerler
  - Dynamic memory yönetimi için uygun pointer'lar tanımlanabilir.

```
// İşaretçi ile tanımlama
int32_t *Numbers;

// İşletim sisteminden bellek talep edebilir
calloc(23, sizeof(int32_t));
//veya
malloc(23, *sizeof(int32_t));
```

  - İşletim sisteminin ayrılan bellek alanını doğrudan dolduramayız. Referans olarak tanımlayabiliriz.

```
int32_t NumberofElement = 10;
int32_t *Numbers;

Numbers = calloc(NumberofElement, sizeof(int32_t));
//veya
Numbers = malloc(NumberofElement, *sizeof(int32_t));
```

  - İşletim sisteminin bellek kapasitesi yeterli gelmez ise, **0** veya **NULL** olarak işlem yapar.
  - **Neden ayırırız?**
    - Program çalıştırılmadan önce ne kadar hafıza gerektiğini bilmediğimizde,
    - Bellek alanı üst sınırı olmayan veri yapıları istendiğinirse,
    - Bellek alanında verimlilik sağlanması istenirse,
    - **Liste eklenme-silme,** adreslerin manipülasyonu, **static** olarak ayrılmış belleğe göre daha kolay yapılır.
    - Programda yapılar ve bağlantılı liste kavramları kullanıldığında dinamik bellek yazılması zorunlu durumdur.

```
// Dynamic olarak ayrılmış bellekler
int *ptr1 = new int;
int *ptr2 = new int[10];

// Dynamic olarak ayrılmış bellek serbest bırakıldı.
delete ptr1;
delete [] ptr2;
```

- **Static Memory Allocation**
  - Bir nesnenin **statik depolama sınıfıyla** bir bağlantısı olduğunu ya da **statik tanımlandığı** varsayılmaktadır.
  - Program çalıştırılmadan önce **1 kez çalışır** ve program ömrü kadar devam eder.
  - **Global** ve **static** değişken örnek olarak verilebilir.
  - **Daha az verimlidir.**
  - Hafızanın yeniden kullanılabilirliği yoktur.
  - **Hızlı işlem yapmaktadır.**
  - **Değişkenler kalıcı olarak tahsis edilir.**
  - **Belleğin, yeniden ayrılma-serbest bırakılma olayları söz konusu değildir.**

```c
void stat(void);

int main()
{
   int i;
   for(i=1; i<=3 ; i++)
      stat();
   return 1;
}
void stat(void)
{
   static int x = 0;
   x = x+1;
   printf("x = %d/n", x);
}
```

DYNAMIC MEMORY	STATIC MEMORY
Değişkenler yalnızca program biriminiz aktif hale gelirse tahsis edilir.	Değişkenler bellekte kalıcı olarak tahsis edilir.
Bellek tahsisi, program yürütülürken yapılır	Bellek tahsisi, program çalıştırılmadan önce yapılır
Hafızanın dinamik tahsisi için heap kullanır	Hafızanın statik tahsisi için stack kullanır
Daha verimlidir	Daha az verimlidir
Allocation durumunda belleğin yeniden kullanılabilirliği vardır. Bellek gerekmediğinde serbest bırakılabilir	Allocation durumunda belleğin yeniden kullanılabilirliği yoktur
Bellek tahsisinde, bellek ayrılmadığında bellek boyutu değişebilir	Bellek tahsisinde, tahsis edildikten sonra bellek boyutu değiştirilemez
Burada hafızanın yeniden kullanılmasına izin verir. Kullanıcı gerektiğinde daha fazla bellek ayırabilir. Ayrıca kullanıcı ihtiyaç duyarsa belleği serbest bırakabilir.	Statik olarak tanımlanan sistemde kullanılmayan belleği yeniden kullanamayız.
Bu bellek ayırma şemasında, yürütme, statik bellek ayırmadan daha yavaştır.	Bu bellek ayırma düzeninde yürütme, dinamik bellek ayırmadan daha hızlıdır
Bu hafızada çalışma zamanında tahsis edilir.	Bu hafızada derleme zamanında tahsis edilir.
Bu ayrılmış bellek program sırasında herhangi bir zamanda serbest bırakılabilir.	Bu ayrılmış bellek programın başından sonuna kadar kalır.
Bu dinamik bellek tahsisi genellikle bağlantılı liste için kullanılır.	Bu statik bellek tahsisi genellikle dizi için kullanılır.