**Dinamik bellek kullanımı**

* [malloc() fonksiyonu](https://www.bilgigunlugum.net/prog/cprog/c_dinamik#c_dinamik01)
* [calloc() fonksiyonu](https://www.bilgigunlugum.net/prog/cprog/c_dinamik#c_dinamik02)
* [realloc() fonksiyonu](https://www.bilgigunlugum.net/prog/cprog/c_dinamik#c_dinamik03)
* [2 boyutlu dizilere dinamik bellek tahsisi](https://www.bilgigunlugum.net/prog/cprog/c_dinamik#c_dinamik04)

C'de daha kaliteli uygulamalar geliştirmek için dinamik bellek kullanımını etkin bir şekilde kullanmamız gerekmektedir. Program normal koşullarda ihtiyaç duyulan bellek tahsisini ve bellek boşaltma işlemlerini işlem satırlarında yer alan kodlara uygun olarak yapar. Ancak, bazı durumlarda, değişkenlere atanacak verilerin boyutu ve ihtiyaç duyulan bellek miktarı programın çalışması esnasında belirlenebileceğinden, bellek tahsis ve boşaltma işlemlerini program çalışması esnasında dinamik olarak gerçekleştirmek gerekir.

Dizi tanımlaması gibi yapılan bazı işlemler belleğin otomatik olarak tahsis edilmesini sağlarken, bazı işlemler bu olanağı sağlamaz. Bu durumda, programın çalışması esnasında gereksinim duyduğumuzda bellek tahsis edebiliriz. Bu işleme dinamik bellek kullanımı yöntemi adı verilir.

Tüm derleyiciler tarafından sağlanan dört adet dinamik bellek tahsis fonksiyonu vardır:

* malloc()
* calloc()
* realloc()
* free()

**malloc() fonksiyonu**

Dinamik bellek kullanımı için genellikle malloc() ve free() fonksiyonları birlikte kullanılır. malloc() fonksiyonu belleği tahsis ederken, free() fonksiyonu ise önceden tahsis edilmiş belleği boşa çıkarır. Bu fonksiyonların genel yapısı aşağıda gösterilmektedir:

void \*malloc (size\_t byte-sayısı);

void free (void \*p);

malloc() fonksiyonunda kullanılan byte-sayısı ifadesi tahsis etmek istediğiniz belleğin byte olarak değerini gösterir. malloc() fonksiyonu tahsis edilmiş belleğin başlangıcını gösteren bir işaretçi geri verir. Tahsis edilmek istenen bellek ihtiyacını karşılayamadığında, NULL bir işaretçi geri verir.

Tahsis edilen belleği boşa çıkarmak için, free() fonksiyonu, daha önce malloc() fonksiyonu ile tahsis edilen belleğin başlangıcını gösteren, bir işaretçi ile kullanılır.

malloc() ve free() fonksiyonları **stdlib.h** başlık dosyasını kullanır.

malloc() fonksiyonunun en önemli özelliği bir işaretçiye bir değer atamadan önce bir değişken adresi atama gereksinimini ortadan kaldırmış olmasıdır. Bunun nedeni, malloc() fonksiyonunun tahsis ettiği belleğin başlangıç adresini otomatik olarak işaretçiye geri vermesidir.

Bir program sona erdiği zaman tahsis edilmiş belleğin tamamı otomatik olarak boşa çıkar.

Şimdi, malloc() ve free() fonksiyonlarının kullanılmasını örneklerle incelemeye çalışalım:

Örnek

#include <stdio.h>

#include <stdlib.h>

int main(void)

{

int \*ip;

// int veri türü boyutu kadar kadar bellek tahsisi

ip = (int\*) malloc(sizeof(int));

\*ip = 126;

printf("Tahsis edilen bellek adres başlangıcı: %p\n", ip);

printf("Tahsis edilen bellekteki değişken değeri: %d", \*ip);

free(ip);

return 0;

}

Yukarıdaki programı derleyip çalıştırdığımızda, aşağıdaki ifadeleri ekrana yazar:

Tahsis edilen bellek adres başlangıcı: 00ec1ab0

Tahsis edilen bellekteki değişken değeri: 126

Program, malloc() fonksiyonunun geri verdiği değeri direkt olarak ip işaretçisine atar. Böylece, int bir değer için ayrılmış olan bellek alanının başlangıç adresi ip işaretçisine atanmış olur. ip işaretçisinin gösterdiği adrese ise 126 sayısını atar. Daha sonra ip işaretçi değeri ile işaretçinin gösterdiği adreste yer alan değeri ekrana yazar. free() fonksiyonu ise tahsis edilen belleği boşaltır.

Bir işaretçiyi kullanmadan önce mutlaka bir adres atanması gerektiğinden, ip işaretçisine bellek adresini malloc() fonksiyonu atamaktadır.

Örnek

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main(void)

{

char \*cp;

cp = (char\*) malloc(40);

if (!cp) {

printf("Bellek tahsis hatası!");

exit(1);

}

strcpy(cp, "Bilgisayar");

printf("%s", cp);

free(cp);

return 0;

}

Yukarıdaki programı derleyip çalıştırdığımızda, aşağıdaki ifadeleri ekrana yazar:

Bilgisayar

Program, 40 byte'lık bir bellek tahsisi yapar ve bu belleğin başlangıç adresini bir char işaretçiye atar. cp işaretçisine bi rkarakter dizisi kopyalar ve diziyi ekrana yazar. Sonra, free() fonksiyonunu kullanarak tahsis edilen belleği boşaltır.

Örnek

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main(void)

{

char \*cp1 = "İlk karakter dizisi";

char \*cp2, \*cp3;

char cdizi[30];

cp2 = "İkinci karakter dizisi";

cp3 = (char\*) malloc(50);

strcpy(cp3, "Üçüncü karakter dizisi"); // malloc() ve free() fonksiyonları kullanılmadığında hata verir.

strcpy (cdizi, "Dördüncü karakter dizisi");

printf("%s\n%s\n%s\n%s", cp1, cp2, cp3, cdizi);

free(cp3);

return 0;

}

Yukarıdaki programı derleyip çalıştırdığımızda, aşağıdaki ifadeleri ekrana yazar:

İlk karakter dizisi

İkinci karakter dizisi

Üçüncü karakter dizisi

Dördüncü karakter dizisi

**calloc() fonksiyonu**

Bellek tahsislerinde calloc() fonksiyonu da kullanılabilir. Bu fonksiyonun malloc() fonksiyonundan tek farkı ayrılacak bellek miktarının eleman boyutu ve eleman sayısı olarak iki argüman halinde tanımlanmış olmasıdır. calloc() fonksiyonunun genel yapısı aşağıda gösterilmektedir:

void \*calloc (size\_t elem-say, size\_t elem\_boy);

Yukarıdaki satırda, elem\_say ifadesi eleman sayısını, elem\_boy ifadesi ise, eleman boyutunu ifade eder. Aşağıda yer alan her iki işlem satırı aynı işlemi gerçekleştirir:

ip = malloc (10 \* sizeof(int));

ip = calloc (10, sizeof (int));

Şimdi, calloc() fonksiyonunun kullanılmasını bir örnek ile incelemeye çalışalım:

Örnek

#include <stdio.h>

#include <stdlib.h>

int main(void)

{

int \*ip, id;

ip = (int \*) calloc(5, sizeof(int));

for (id=0; id<5; id++) {

\*(ip+id) = (id+1) \* 10;

printf("%p adresindeki değer: %d\n", (ip+id), \*(ip+id));

}

free(ip);

return 0;

}

Yukarıdaki programı derleyip çalıştırdığımızda, aşağıdaki ifadeleri ekrana yazar:

00180538 adresindeki değer: 10

0018053c adresindeki değer: 20

00180540 adresindeki değer: 30

00180544 adresindeki değer: 40

00180548 adresindeki değer: 50

Program, calloc() fonksiyonunu kullanarak 5 adet int değer için bellek tahsisi yapar. Her bellek adresine int bir değer atar. Daha sonra, bellek adreslerini ve bu adreslere atadığı değerleri ekrana yazar.

**realloc() fonksiyonu**

realloc() fonksiyonu, daha önce malloc(), calloc() veya realloc() fonksiyonu ile tahsis edilen belleğin boyutunu, boyutu byte olarak ifade edilen size parametre değeri kadar, değiştirir.

void\* realloc(void \*ptr, size\_t size);

Yeniden bellek tahsis işlemi ptr parametresi ile gösterilen bellek adresi büyütülerek veya küçültülerek yapılabilir. Bellek alanı genişletilirse, tahsis edilen önceki bellek içeriği değişmeden kalır ve eklenen bellek içeriğine herhangi bir değer atanmaz.

Yeterli bellek yoksa, eski bellek bloğu serbest bırakılmaz ve NULL bir işaretçi geri döndürülür.

**ptr** parametresi yeniden tahsis edilecek bellek bölgesini ve **size** parametresi tahsis edilecek belleğin yeni boyutunu byte olarak gösterir.

Başarı durumunda tahsis edilen belleğin başlangıç adresini geri döndürür. Bellek sorunlarını engellemek için, gerekli işlemler yapıldıktan sonra, bu işaretçi free() veya realloc() fonksiyonu ile boşaltılmalıdır. Hata durumunda, NULL bir işaretçi geri döndürülür. Bu durumda, geçerliliğini devam ettiren önceki işaretçi free() veya realloc() fonksiyonu ile boşaltılmalıdır.

Örnek

#include <stdio.h>

#include <stdlib.h>

int main(void)

{

int \*ip, id;

ip = (int \*) malloc(5 \* sizeof(int));

for (id=0; id<5; id++) {

\*(ip+id) = id+1;

printf("%p adresindeki değer: %d\n", (ip+id), \*(ip+id));

}

ip = (int \*) realloc(ip, 10 \* sizeof(int));

printf("Genişletilmiş bellek değerleri:\n");

for ( ; id<10; id++) { // Burada id değişken değeri 5 olarak başlar.

\*(ip+id) = id+1;

printf("%p adresindeki değer: %d\n", (ip+id), \*(ip+id));

}

free(ip);

return 0;

}

Yukarıdaki programı derleyip çalıştırdığımızda, aşağıdaki ifadeleri ekrana yazar:

00de1690 adresindeki değer: 1

00de1694 adresindeki değer: 2

00de1698 adresindeki değer: 3

00de169c adresindeki değer: 4

00de16a0 adresindeki değer: 5

Genişletilmiş bellek değerleri:

00de16a4 adresindeki değer: 6

00de16a8 adresindeki değer: 7

00de16ac adresindeki değer: 8

00de16b0 adresindeki değer: 9

00de16b4 adresindeki değer: 10

Program, malloc() fonksiyonunu kullanarak 5 adet int değer için bellek tahsisi yapar ve her bellek adresine int bir değer atayarak bellek adreslerini ve bu adreslere atadığı değerleri ekrana yazar. Daha sonra, realloc() fonksiyonuyla tahsis edilen bellek miktarını 10 int değer alacak kadar genişletir ve yeni atadığı değerleri ve bellek adreslerini ekrana yazar.

**2 boyutlu dizilere dinamik bellek tahsisi**

C'de, işaretçi dizileri kullanarak çok boyutlu dizilere dinamik bellek tahsisi yapabiliriz. Şimdi, bir örnek üzerinde iki boyutlu bir karakter dizisine bellek tahsisi işlemini incelemeye çalışalım:

Örnek

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main(void)

{

char \*pdizi[5];

int id;

// Bellek tahsisi

for (id=0; id<5; id++) {

pdizi[id] = (char \*) malloc(20 \* sizeof(char));

}

// Dizi atama

for (id=0; id<5; id++) {

strcpy(pdizi[id], "Karakter dizisi");

}

// Yazdırma

for (id=0; id<5; id++) {

printf("%s\n", pdizi[id]);

}

// Bellek boşaltma

for (id=0; id<5; id++) {

free(pdizi[id]);

}

return 0;

}

Yukarıdaki programı derleyip çalıştırdığımızda, aşağıdaki ifadeleri ekrana yazar:

Karakter dizisi

Karakter dizisi

Karakter dizisi

Karakter dizisi

Karakter dizisi

Program 5 elemanlı bir işaretçi dizisi oluşturur. Her bir işaretçi dizisine 20 byte boyutundaki belleğin başlangıç adresini atar. İşaretçi indeksleme yöntemi ile her bir işaretçiye "Karakter dizisi" ifadesini kopyalar ve ekrana yazdırır. Program sona ermeden önce, free() fonksiyonunu bir döngü içinde kullanarak tahsis edilen belleği boşaltır.