Veri Yapıları ve Algoritmalar

Dinamik Bellek Yönetimi

Öğr. Gör. M. Ozan AKI

r1.0

Neden

Değişken ve Dizilerde tanımlanarak bellekten ayrılan sabit bellek alanları büyük çoğunlukla sorunsuz bir şekilde çalışır ve kullanılılır.

Ancak bazı durumlarda, çalışma zamanında bellek ihtiyacı artabilir ya da azalabilir.

Bu gibi durumlarda, çalışma zamanında değişken eklenip çıkarılamayacağı ve sabit dizilerin boyutu değiştirilemeyeceği için dinamik bellek kullanılır.

Neden

Örneğin, büyükçe bir dosya belleğe okunarak bir işlem yapılacaksa, sabit bir büyüklük ne kadar olmalıdır? Dosya ne kadar büyük ? Belleğimiz ne kadar ?

Program içerisinde çok büyük bir dizi tanımlarım, bütün dosyaları açarım → Bilgisayarın belleğinde yeterli yer yoksa programınız hiç çalışmayacaktır.

Makul bir dizi boyutu tanımlarım böylece bellek azda olsa program çalışır → Fakat dosya boyutu dizi boyutundan büyükse program çalışır ancak dosyayı açamaz.

malloc ve free

C dili dinamik bellek kullanımı için iki kütüphane fonksiyonu tanımlar:

void* malloc(int size);

parametre olarak geçilen size kadar **byte** heap bellekten yer alır ve başlangıç adresine bir pointer döndürür. Yer ayrılamaz ise NULL döndürür.

free(void* ptr);

Daha önce malloc ile ayrılmış bellek alanının başlangıç adresi parametre olarak verilir. Bu bellek alanı serbest bırakılır.

dinamik diziler

C dilindeki herhangi veri tipinde tanımlanan bir pointer için malloc ile istenilen boyutta (sayida!) yer ayırmak için

```
int *iptr;
iptr = (int*)malloc(sizeof(int)*120);
...
free(iptr);
```

malloc dönüş işaretçisi mutlaka (cast) ile pointer değişkenine uygun tipe dönüştürülmelidir.
Ayrılan bellek alanları byte cinsinden olduğundan, dizideki eleman sayısı için, veri tipinin kapladığı alan,

sizeof(tip) * dizide istenen eleman sayısı

olarak parametre geçilmelidir.

dikkat!

- Pointer hataları derleyici tarafından yakalanamaz, ancak çalışma zamanında ortaya çıkar!
- Değer atanmamış bir pointer kullanmayın!
- malloc ile aldığınız belleğin adresini kaybetmeyin!
- İşi biten bellekleri serbest bırakmayı unutmayın! (C Dilince dağıttıklarınızı arkanızdan toplayan bir **code garbage yoktur**!)
- İşi biten ve serbest bırakılan pointerlara hemen NULL değerini atayın. **free() bunu yapmaz**!

realloc()

- Daha önceden alınmış bir belleğin verilerini kaybetmeden boyutunu büyütmek ya da küçültmek gerekebilir.
- Bu gibi durumda iki seçenek vardır;
  - **malloc() ile istenen boyutta yeni bir bellek alınır**
  - **eski bellekteki veriler yeni belleğe kopyalanır,**
  - **eski bellek serbest bırakılır.**
- Ya da;
  - Tüm bunları otomatik olarak yapan **realloc()** kullanılır.

realloc()

```
int *p;

p = (int*)malloc(sizeof(int)*100);
...

int *temp;
temp = (int*)realloc( p, sizeof(int)*150);
if(temp!=NULL)
{
  p = temp;
}
```

string.h

void* memset(void *dest, int c, size_t *count*);

void* memcpy(void *dest, void *src, size_t *count*);

void* memmove(void *dest, void *src, size_t *count*);

int strcmp(const char **str1*, const char **str2*);

char* strcpy(char **str1*, const char **str2*);

size_t strlen(const char **str*);

char* strstr(const char **str1*, const char **str2*);