

Bölüm 12
ÜST DÜZEY KONULAR

Dinamik Bellek Ayırma

Önişlemci

Değişken Depolama Tipleri

Komut Satırı Parametreleri

Fonksiyon Gösteren Göstergeler

Dinamik Bellek Ayırma

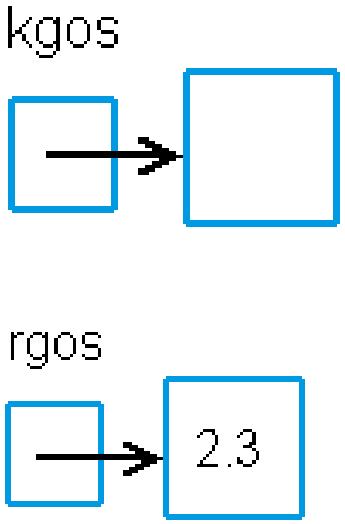
Yürütme sırasında bellekte yer ayrılan veri yapılarına **dinamik veri yapısı**, dinamik değişkenler için bellekte ayrılan yere **alt yığın** adı verilir.

`malloc()` fonksiyonu, alt yığından bir veri tipi kadar (örneğin tamsayı için 4 bayt) yer ayırır ve ayırdığı yerin adresini geri döndürür.

```
malloc(sizeof(veri_ tipi))
```

Örnek:

```
char *kgos;
float *rgos;
kgos = (char*)malloc(sizeof(char));
rgos = (float*)malloc(sizeof(float));
*rgos = 2.3;
```

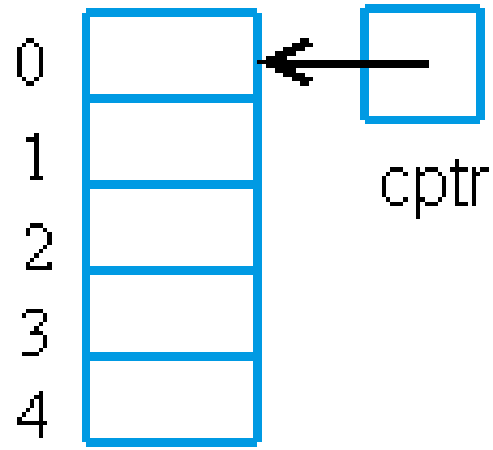


Dinamik Bellek Ayırma

`calloc()` fonksiyonu bir dizi için alt yığından belirtilen eleman sayısı ve eleman tipinde yer ayırır ve dizinin başlangıç adresini geri döndürür.

Örnek:

```
int *cptr;
cptr =(int*)calloc(x,sizeof(int));
```



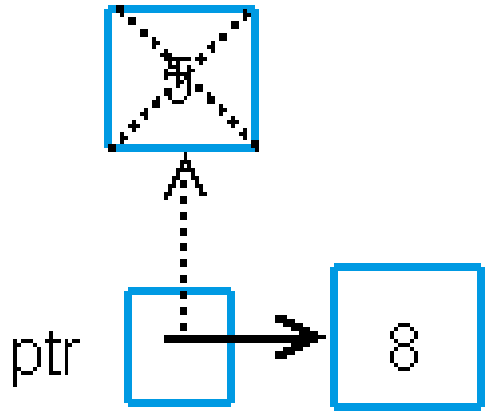
Dinamik Bellek Ayırma

`free()` fonksiyonu dinamik olarak yaratılan hücreleri salıvermek için kullanılır.

```
free(gösterge_adı);
```

Örnek:

```
int *ptr;
ptr = (int*) malloc (sizeof(int));
*ptr = 5;
...
free(ptr);
ptr = (int*) malloc (sizeof(int));
*ptr = 8;
```



ÖnİşlemciÖnİşlemci

Önişlemci bir C programı derlenmeden önce, programı hazırlamakla görevlidir.

Örnek:

```
#define SAYI 50
int main(void)
{
   int x = SAYI;
   printf("%d", SAYI);
}
```

ÖnİşlemciMacrolar çok kullanılan komut veya işlemlere isim vermek amacıyla kullanılır.

Örnek:

```
#define ALAN(en,boy)  ((en) * (boy))
int main(void)
{  int a;
   a = ALAN(2,3);
}
```

Makro açılımı gerçekleştiğinde, program metni aşağıdaki şekli alır.

```
int main(void)
{  int a;
    a = ((2)*(3));
}
```

ÖnİşlemciÖnİşlemci

Özel Kütüphane Dosyaları tırnak işaretleri "..." arasında tanımlanan, programcı tarafından hazırlanmış özel başlık dosyalarıdır

#include "*özel_dosya*"

Örnek:

```
#include <stdio.h>
int x;
void fon() {printf("%d",x);}
```



"**bas.h**" dosyası

```
#include "bas.h"
      int main(void)
      { x=5;
        fon();
      }
```



program

Değişken Depolama Tipleri

- **`auto`** değişkeni, fonksiyon parametreleri ve yerel değişkenlerin depolanma şeklidir. Bu tip değişkenler fonksiyon çağırıldığında yaratılıp, fonksiyon sona erdiğinde yok edilirler.

- **genel** değişkenler program başladığında yaratılan, programın her yerinde, aynı isim yeniden tanımlanmadıkça kullanılabilen ve program bitiminde yok olan değişkenlerdir.

- **`extern`** tanımı fonksiyon isimlerine ve değişkenlere farklı dosyalardan erişimi sağlar.

- **`static`** değişkenlerin kapsamı sadece tanımlandıkları fonksiyondur, ancak bu tip değişkenlerin ömrü programın başından sonuna kadardır.

- **`register`** değişkenleri çok fazla kullanılacak olan ve bu nedenle hızlı erişimi olması gereken değişkenlerdir.

Komut Satırı Parametreleri

Özellikle DOS ve Unix-tabanlı sistemlerde C programının **`main()`** fonksiyonuna dışarıdan parametre gönderebiliriz. **`main()`** fonksiyonunda tanımlanmış iki parametre yer alır.

```
int main (int argc, char *argv[]);
```

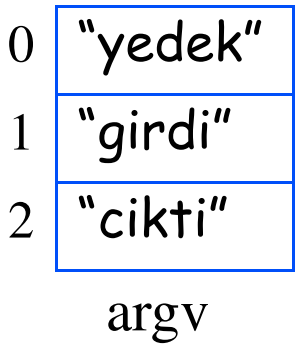
`argc`: toplam parametre sayısı
`argv`: parametre adlarından oluşan dizgi dizisi

`Örnek:` İki parametre alan **`yedek`** adlı bir program yazalım. Bu programı çalıştırmak için Unix veya DOS tabanlı sistemlerde aşağıda verilen komut satırını yazmamız gerekir.

```
yedek girdi cikti
```

Aşağıda verilen program komutları **`girdi`** dosyasını açar.

```
FILE *dos1;
dos1 = fopen(argv[1],"r");
```



Fonksiyon Gösteren Göstergeler

Bir göstergeye bir fonksiyonun adresi atanarak, o fonksiyon parametre olarak başka bir fonksiyona yollanabilir veya bir dizinin elemanı olarak tanımlanabilir

Örnek:

```
int (*fon)(char, float)


void fpar( int (*param)(int,int)) ;
```