INF114 Algoritma ve Ileri Programlama

Dinamik Bellek Yönetimi 03/04/2023

Dinamik Bellek İşlemleri

```
void *malloc(size_t size);
```

Bellekte size kadar bayt yer ayırır. Ayrılan yer sıfırlanmaz. Başarılı olduğunda ayrılan yere (void) **POINTER** (gösterici), başarısız olduğunda NULL döndürür. Yani geri dönüş değeri olan **POINTER** bellekte malloc ile açılan size kadar baytlık yerin adresini gösterir. Ayrıca fonksiyonun void * dönüşlü olarak tanımlanması type-casting yapmaya gerek olmadığı anlamına gelir.

```
void *calloc(size_t nmemb, size_t size);
```

Her biri size boyutunda, nmemb adet eleman için yer ayırır ve ayrılan yeri sıfırlar. Başarılı olduğunda ayrılan yere pointer, başarısız olduğunda NULL döndürür.

```
void *realloc(void *ptr, size_t size);
```

ptr pointer'ı ile gösterilen bellek bloğunun boyutunu, size olacak şekilde değiştirir (küçültür/büyütür). Başarılı olduğunda yeni bloğa pointer, başarısız olduğunda NULL döndürür.

```
void free(void*ptr);
```

Daha önce ayrılan ve adresi ptr'de saklanan bellek alanını boşaltır.

Bağlı Liste Eleman Ekleme Örneği

```
void insert_element(struct node** head, int data) {
    // **head elemani baslangic elemanini gosteren pointer'↩
       in adresini tutar. Yani head'i gosteren pointer'i ↩
       gosterir (pointerin pointeri). Buna void ↩
       fonksiyonda neden ihtiyac bulunuyor? Cunku ↩
       fonksiyona verilen parametrelerin main'deki ↩
       adreslerinde degil bellekte baska bir lokal gozde ↩
       islem yapilir (parametrenin main'deki adresi ↩
```

```
    verilmedigi surece). Biz ise dogrudan verilen elemanin bellek gozunde islem yapmak istiyoruz. Bunun nedeni ise bagli liste bos oldugunda veya en basa eleman atanacaginda head elemaninin gosterdigi adres degisir. Yani head pointer'inin bellek gozunde yazan adres degerini degistirmemiz gerekir. Bunu (void fonksiyonda) yapabilmemiz icin head'i gosteren pointer'in main'deki bellek adresini bilmemiz ve o adres gozunde gosterdigi yeni elemanin adresini guncellemeliyiz. Bu nedenle biz head pointer'in main'deki adresini yani **head'i parametre olarak fonksiyona veririz.

struct node* cur = (*head);
// Tanimladigimiz cur pointer'i bagli listenin ilk elemani olan head'in gosterdigi bellegin adresini gosterecek sekilde atama yapilir.

if(cur == NULL) {
// Eger bagli listede eleman yoksa,
    cur = malloc(sizeof(struct node));
    cur->data = data;
    cur->next = NULL;
// (1) yeni eleman icin yer tahsis edilir ve pointeri dondurulur.
    *head = cur;
// (2) head'in gosterdigi adres, cur'in gosterdigi adres olarak guncellenir. Tekrardan hatirlamak gerekirse **head parametre olarak alinip, main'de eleman ekleme fonksiyonu head pointer'inin adresi verilip cagrildiginde dogrudan head pointer'inin yeni gosterecegi adresi kalici olarak guncelleyebiliriz. Sonuc olarak head pointer'i artik bellekte yeni tahsis edilen elemanin adresini gosterir.
} else {
    // Bagli listede en azindan bir eleman bulunuyorsa
    struct node* nxt = (*head)->next;
    struct node* tmp = malloc(sizeof(struct node));
    tmp->data = data;

    // Eklenecek eleman yeni head olacak ise
    if(data < cur->data) {
        tmp->next = cur;
        *head = tmp;
    } else {
        while(nxt != NULL) {
            // Eklenecek eleman arada bir yere eklenecek ise
            if(data >= cur->data && data < nxt->data) {
```

```
                    cur->next = tmp;
                    tmp->next = nxt;
                    return;
                }
                cur = nxt;
                nxt = nxt->next;
            }
            // Eklenecek eleman son eleman olacak ise
            if(data >= cur->data) {
                tmp->next = NULL;
                cur->next = tmp;
            }
        }
    }
}
```