Ders 41

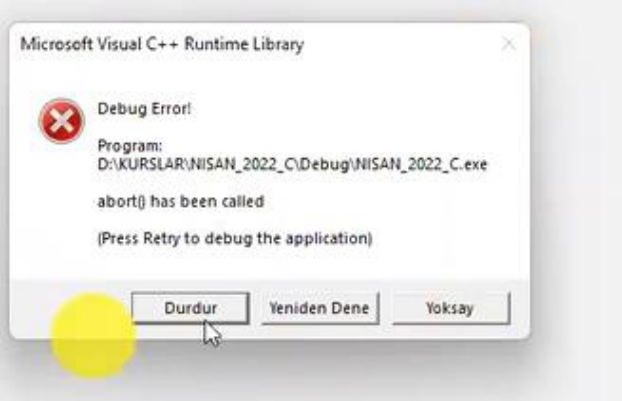
Örnek f1 f2 çağırma abort etmek.

```c
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

void f2(void)
{
    printf("f2 cagrildi\n");
    abort();
    printf("f2 sona erdi\n");
}

void f1(void)
{
    printf("f1 cagrildi\n");
    f2();
    printf("f1 sona erdi\n");
}

int main(void)
{
    printf("main cagrildi\n");
    f1();
    printf("main sona erdi\n");
}
```



F2 içinde abort olduğu için abort ekrana yazı yazdırdı benim yüzümden prigram sonladı diye

Ama f1 ve f1 çağırıldı yazısı çıktıktan sonra sonlanma olacak.

//

Dinamik bellek yönetimi

//

Örnek malloc kullanımı

```
{
    int n;

    printf("kac tam sayi: ");
    scanf("%d", &n);

    int* pd = (int *)malloc(n * sizeof(int));
    if (!pd) {
        fprintf(stderr, "cannot allocate memory!\n");
        exit(EXIT_FAILURE);
    }

    randomize();
    set_array_random(pd, n);
    print_array(pd, n);
    //code
```

İlk önce ihtiyacım olan bellek alanını malloc kullanarak aldım

Burada int dizi gibi olmasını istediğim için int *p ye adresi verdim ve int* türüne dönüşüm yaptım çünkü malloc void* türünde geri dönüş değeri üretiyor.

Sonra fonksiyonun başarı bilgisini sınamak için if ile bir koşul oluşturdum malloc başarısızssa program bitsin

Diziler için kullandığımız fonksiyonları burdada kullanabiliriz. Çünkü bu fonksiyonlar elemanlar sıralımı diye bakıyordu.

Birde malloc verdiği byte ları çöp değeri ile veriyor.

Bellekle işim bitince free() ile belleği boşaltıcam

Linux idiomu if içinde atama değeri

```
printf("kac tam sayi: ");
scanf("%d", &n);
int* pd;

if ((pd = (int *)malloc(n * sizeof(int)) == NULL)) {
    fprintf(stderr, "cannot allocate memory!\n");
    exit(EXIT_FAILURE);
}

randomize();
set_array_random(pd, n);
print_array(pd, n);

//code
free(pd);
//..
```

Burada atama operatörünün ürettiği değerden faydalandık.

Örnek checked malloc kullanılan yapıyı fonksiyon yazmak.

```
void* checked_malloc(size_t n)
{
    void* vp = malloc(_Size:n);
    if (!vp) {
        fprintf(_Stream:stderr, _Format:"cannot allocate memory!\n");
        exit(_Code:EXIT_FAILURE);
    }

    return vp;
}
```

Her seferinde aynı kodları yazıp kontrol eleceğime bir kere fonksiyon yazdık.

//

Örnek en tipik free hataları.

1. free fonksiyonu çağrıldığında deallocate edilen bellek bloğunun adresi değil
bu pointer'i dereference etmek ta

```
    scanf("%d", &n);
    int* pd;

    if ((pd = (int *)malloc(n * sizeof(int)) == NULL)) {
        fprintf(stderr, "cannot allocate memory!\n");
        exit(EXIT_FAILURE);
    }
    int* p_ar = pd;
    randomize();
    set_array_random(pd, n);
    print_array(pd, n);
    free(pd);

    print_array(p_ar, n); //ub
```

Bu kodda p_ar artık dandling pointer gösterdiği değerde artık kullanılan bir bellek yok eğer bunu çağırıp kullanırsak veya *direference edersek tanımsız davranış olur en çok yapılan hata

//

2. free ile edinilmiş bellek bloğunun bir kısmını geri vermek mümkün değil

```
{
    int n;

    printf("kac tam sayi: ");
    scanf("%d", &n);
    int* pd;

    if ((pd = (int *)malloc(n * sizeof(int)) == NULL)) {
        fprintf(stderr, "cannot allocate memory!\n");
        exit(EXIT_FAILURE);
    }

    ///

    free(pd + n / 2)

}
```

Burada kullanıcı bazen pd yi dizinin bir kısmını deallocet yapmak için pointer aritmetiği kullanmayı deniyor bu kesinlikle bir tanımsızdavranış.

//

3. static olarak (derleyicinin yerini ayırdığı) bellek bloklarının adresleri ile free işlevini çağırmayın

```
//no operations
int main(void)
{
    int a[100];
    ///

    int* pd = a;

    ///...


    free(pd);

}
```

Dinamik olarak oluşturulmamış bellek bloğunun adresini free ile çağırmak tanımsız davranış

//

```
4.I double deletion
```

Serbest bırakılan bellek adresini bir daha free fonksiyonuna gönderilemez.

```
int main(void)
{
    int n;

    printf("kac tam sayi: ");
    scanf("%d", &n);
    int* pd;

    if ((pd = (int *)malloc(n * sizeof(int)) == NULL)) {
        fprintf(stderr, "cannot allocate memory!\n");
        exit(EXIT_FAILURE);
    }

    int* px = pd;
    ///

    free(pd);

    ///
    free(px);

}
```

Burada kullanılan bir bellek bloğunu zaten free ileserbest bıraktık ancak ayı bloğu gösteren farklı birpointer ile tekrar free çağıramayız bu tanımsız davranış olur.

//

5. free fonksiyonunu hiç çağırmamak memory lick olur.

```
pd = malloc(???);

pd = &x;
```

Burada belleği gösteren pointera başka değer atanmış bu yüzden artık free ile bellek deallocet yapılamıyacak

//

6. free yi null ile çağırmak

```
void free(void *vptr)
{
    if (!vptr)
        return;
}
```

Bu durmda fonksiyon hiçbir iş yapmaz tanımsız değil

//