Kendi header dosyalarımızı yazmak

Bu aşamaya kadar çeşitli .h dosyalarını kodumuza include ettik. Şimdi sıra kendi header dosyalarımızı yazmakta.

Fonksiyonlar ünitesinde header dosyalarının fonksiyon prototiplerini içerdiğini görmüştük. Header dosyaları prototipleri tutmakla beraber fonksiyonların body kısımlarını tutmaz. Yani bizler hem .h hem de .c uzantılı dosyalar yazacağız.

Diyelim ki bir karenin kenarını alıp çevre ve alan bulan iki adet fonksiyon için header file yazacağız.

```
int alan(int kenar){
      return kenar * kenar;
}

int cevre(int kenar){
      return kenar * 4;
}
```

Fonksiyonlarımız bunlar olsun.

Bu fonksiyonların prototiplerini geometri.h ve fonksiyonların kendilerini geometri.c isimli dosyalara koyalım

geometri.h dosyası:

```
int alan(int kenar);

int cevre(int kenar);
```

geometri.c dosyası

```
int alan(int kenar){
      return kenar * kenar;
}

int cevre(int kenar){
      return kenar * 4;
}
```

Şimdi ise bu fonksiyonları bir main.c dosyasında kullanmaya çalışalım:

```c
#include <stdio.h>
#include "geometri.h"

int main(){

    int kenar = 5;

    int Alan = alan(kenar);
    int Cevre = cevre(kenar);

    printf("%d %d\n", Alan, Cevre);
    return 0;
}
```

Kodda gördüğünüz gibi geometri.h dosyasını çift tırnak içinde include(dahil) ettik. Bu compilerın, linkleme esnasında bu dosyayı bizim kendimizin yazdığını anlamasını sağlar.

Şimdi bunu çalıştıralım: şu ana kadar yaptıklarımızdan farklı olarak sadece main dosyasını değil, yazdığımız geometri.c dosyasını da derlemeliyiz. Bunun için:

gcc main.c geometri.c -o main

komutunu terminale giriyoruz, “main.c ve geometri.c dosyalarını derleyip linkledikten sonra bana main isminde bir dosya ver” diyoruz.

Main dosyasını çalıştırırken bir değişiklik yok:

./main komutuyla programımızı çalıştırıyoruz.

Şimdi gelelim iki tane ayrıntıya:

1)diyelim ki bu ffonksiyonların kendi içinde alan ve çevre değerlerini bastırmasını istiyoruz. Yani değişiklik sonucunda geometri.c şu şekilde görünecek:

```c
int alan(int kenar){
      printf("%d\n",kenar * kenar);
      return kenar * kenar;
}

int cevre(int kenar){
      printf("%d\n", kenar * 4);
      return kenar * 4;
}
```

Printf() fonksiyonunu kullanmamız için stdio.h dosyasını include etmemiz gerekiyor. main fonksiyonunda dahil ettik ama geometri.c dosyası ve main.c dosyaları ayrı ayrı derleneceği için geometri.c içine tekrardan stdio.h header dosyasını include etmeliyiz. Sonuç olarak geometri.h dosyası şu şekilde görünecek

```
#include <stdio.h>

int alan(int kenar){
      printf("%d\n",kenar * kenar);
      return kenar * kenar;
}

int cevre(int kenar){
      printf("%d\n", kenar * 4);
      return kenar * 4;
}
```

Şimdi ise structların işin içine girdiği durumlar için bir örnek yapalım:
Diyelim ki, Diktortgen isimli struct uzun ve kısa kenar için iki adet elemana sahip olsun. Diktörtgenin çevresini ve alanını bastıracak 2 adet fonksiyon için .h ve .c dosyaları yazalım:

geometri.h dosyası şu şekilde görünecek:

```
struct Dikdortgen{
    int uzun;
    int kisa;
};

void alan(struct Dikdortgen d);
void cevre(struct Dikdortgen d);
```

struct da bir tanımlama olduğu için .h dosyasında duruyor. Bir önceki örnekten farklı olarak bu header dosyasını da geometri.c içine include ediyoruz çünkü tanımladığımız structı fonksiyonlarda kullanmamız gerekecek. Bir önceki örnekte bunu yapmamıza gerek kalmamıştı çünkü kullandığımız değişkenleri int idi yani biz tanımlamamıştık, c dilinde zaten var olan değişkenlerdi.

geometri.c dosyası:

```c
#include <stdio.h>
#include "geometri.h"

void alan(struct Dikdortgen d){
    printf("%d\n",d.kisa * d.uzun);
}

void cevre(struct Dikdortgen d){
    printf("%d\n",  2 * (d.uzun + d.kisa) );
}
```

Son olarak main dosyasında da geometri.h dosyasını dahil ediyoruz çünkü hem fonksiyonları kullanmamız lazım hem de tanımladığımız structı sadece geometri.h içinde tanımladık. Structı main dosyasında kullanmak için geometri.h dosyasını include etmemiz gerekiyor.

main.c dosyası:

```c
#include <stdio.h>
#include "geometri.h"

int main(){

    struct Dikdortgen dikdortgen = {5 , 3};

    alan(dikdortgen);
    cevre(dikdortgen);

    return 0;
}
```