

4.Ders : C Özyineleme (Recursion in C)

Bu derste, C programlamada özyinelemeli bir fonksiyon yaratmayı öğreneceksiniz.

İçindekiler :

özyineleme

Özyineleme nasıl çalışır?

Örnek: Özyinelemeyi Kullanan Doğal Sayıların Toplamı

Özyinelemenin Avantajları ve Dezavantajları

Kendisini çağıran bir fonksiyon özyinelemeli fonksiyon olarak bilinir. Ve bu teknik özyineleme olarak bilinir.

Özyineleme nasıl çalışır?

```
void recurse()  
{  
    ... ..  
    recurse();  
    ... ..  
}
```

```
int main()  
{  
    ... ..  
    recurse();  
    ... ..  
}
```

How does recursion work?

```
void recurse()
{
    ... ..
    recurse();
    ... ..
}

int main()
{
    ... ..
    recurse();
    ... ..
}
```

The diagram shows two function definitions. The first is `void recurse()` with a body containing three lines: `... ..`, `recurse();`, and `... ..`. The second is `int main()` with a body containing three lines: `... ..`, `recurse();`, and `... ..`. A horizontal arrow points from the `recurse();` line in `main()` to the `recurse()` line in the `recurse()` function. A vertical line extends upwards from the `recurse();` line in `main()`, and another vertical line extends downwards from the `recurse();` line in the `recurse()` function. These two vertical lines are connected by a horizontal line, and the entire structure is labeled "recursive call".

Özyineleme, bunu önlemek için bir koşul sağlanana kadar devam eder.

Örnek: Özyinelemeyi Kullanan Doğal Sayıların Toplamı

```
#include <stdio.h>
int sum(int n);

int main()
{
    int number, result;

    printf("pozitif bir tamsayı girin: ");
    scanf("%d", &number);

    result = sum(number);

    printf("Toplam = %d", result);
    return 0;
}
```

```
}  
  
int sum(int num)  
{  
    if (num!=0)  
        return num + sum(num-1); // sum() function calls  
itself  
    else  
        return num;  
}
```

Çıktısı :

```
pozitif bir tamsayı girin:3  
Toplam = 6
```

Başlangıçta, sum() fonksiyonu argüman olarak iletilen sayı ile birlikte main() fonksiyonundan çağrılır.

Farz edelim ki num değeri başlangıçta 3'tür. Bir sonraki işlev çağrısı sırasında 2 , sum() fonksiyonuna iletilir. Bu işlem, num 0'a eşit olana kadar devam eder.

Num 0'a eşit olduğunda, if koşulu başarısız olur ve diğer kısım tamsayıların toplamını main () işlevine döndürerek yürütülür.

```

int main() {
    ... .. 3
    result = sum(number)
    ... ..
}

int sum(int n)
{
    if(n!=0) 3 2
        return n + sum(n-1);
    else
        return n;
}

int sum(int n)
{
    if(n!=0) 2 1
        return n + sum(n-1);
    else
        return;
}

int sum(int n)
{
    if(n!=0) 1 0
        return n + sum(n-1);
    else
        return n;
}

int sum(int n)
{
    if(n!=0)
        return n + sum(n-1);
    else
        return n;
}

```

$3+3 = 6$
is returned

$1+2 = 3$
is returned

$0+1 = 1$
is returned

0
is returned

Özyinelemenin Avantajları ve Dezavantajları

Özyineleme, programı zarif ve daha okunaklı hale getirir. Ancak, projenizde performans daha önemli ise, özyineleme genellikle daha yavaş olduğu için döngüler kullanın.

Her özyinelemenin bir döngü olarak modellenebileceğini unutmayın.

Performansa ihtiyacınız var ise , döngüler kullanın, ancak kod bazen okunması zor ve çirkin görünebilir. Daha şık ve okunabilir koda ihtiyacınız var ise özyinelemeyi kullanın, ancak biraz performansdan ödün veriyorsunuz bunu da unutmayın.

Daha fazla bilgi edinmek için bu örnekleri inceleyin:

[Özyineleme kullanarak Doğal Sayıların Toplamını Bulmak için C Programı](#)

[Özyineleme Kullanarak Sayının Faktörünü Bulmak için C Programı](#)

[Özyineleme kullanarak OBEB bulmak için C programı](#)