Algoritma ve Programlama -1

8. HAFTA

Özyinelemeli (Recursive) ve Hazır Fonksiyonlar

7. Hafta - Tekrar

- Fonksiyonlar
  - Tanımlanması
  - Kullanılması
  - Çağrılması
  - Parametreler ve Geri Dönüş Değerleri

Örnek Proje

- Bir elektrik şirketi müşterilerine ait verilerin kontrolünü sağlayabilmek adına bir otomasyon programı talep etmektedir.

- Programdan istenenler;
  - Her müşteriye ait müşteri id, kayıt yılı, güvence bedeli, tarife, sayaç değeri, ödenmemiş fatura miktarı gibi bilgiler tutulmaktadır.
  - Her müşteri için yeni sayaç değeri girilerek tarifesine göre fatura miktarı hesaplanacaktır.
  - Girilen sayaç değeri öncekinden küçükse hata mesajı verilerek yeni sayaç değeri istenecektir.
  - Müşteriler sistemden ödeme yapabileceklerdir.
  - Sistem ödeme yapmayan müşterileri talep edildiği takdirde gösterecek, güvence bedeli kadar ödeme yapmayan müşteri varsa aboneliği sonlandırılacaktır.
  - Müşterilere ait genel bilgiler istenildiği zaman listenecektir.
  - İnteraktif menü ile program etkili bir biçimde kullanılabilecektir.
  - Sisteme yeni müşteri kaydı yapılabilecektir.

Örnek Proje

- Fatura miktarı hesaplama;
  - Kullanıcı tarafından girilen sayaç değerinden mevcut sayaç değeri çıkarılır.
  - Sayaç farkı ile sayaç başına düşen elektrik birim fiyatı (3 TL) çarpılır.
  - Müşterinin kullandığı tarifeye ve kayıt yılına göre indirim belirlenir.
    - Tarife 1 - %10, Tarife 2 - %8, Tarife 3 - %5, Tarife 4 - %2
    - 2002 yılından önce - %10, 2002-2012 - %5
  - İndirimden sonra bulunan miktar ödenmemiş borç değerine eklenir ve güvence bedeli ile kontrol gerçekleştirilir.
    - Eğer güvence bedelinden daha büyük borç çıkarsa uyarı mesajı verilerek abonelik sonlandırılır.

Örnek Proje

- List() -- Check
- Add() -- Check
- Calculate()
- Pay()
- List_Debtor()

Fonksiyon Örnekler

- Klavyeden girilen sayının armstrong olup olmadığın kontrol eden C kodunu yazınız.
  - Her basamağın, sayının basamak sayısı kadar kuvvetinin alınıp toplanmaları ile elde edilen sonuç **armstrong sayılarını** verir.

- Kullanıcıdan verilen boyuta göre Pascal üçgenini ekranda gösteren C kodunu yazınız.

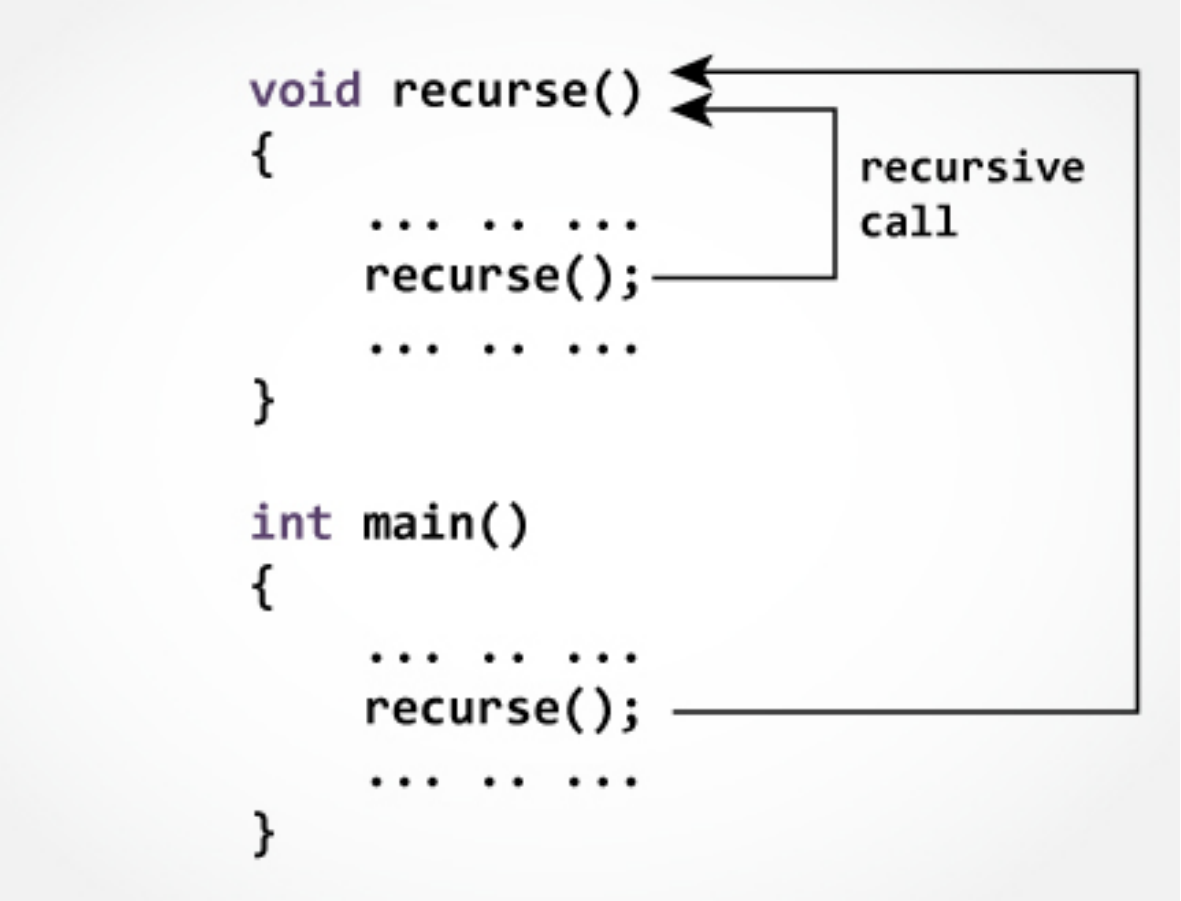
Özyinelemeli (Recursive) Fonksiyonlar

- A function that calls itself is known as a recursive function.
- And, this technique is known as recursion.
- How recursion works?

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

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

How does recursion work?



Özyinelemeli (Recursive) Fonksiyonlar

- The recursion continues until some condition is met to prevent it.

- To prevent infinite recursion, if...else statement (or similar approach) can be used where one branch makes the recursive call and other doesn't.

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

int main()
{
     int number, result;

     printf("Enter a positive integer: ");
     scanf("%d", &number);

     result = sum(number);

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

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

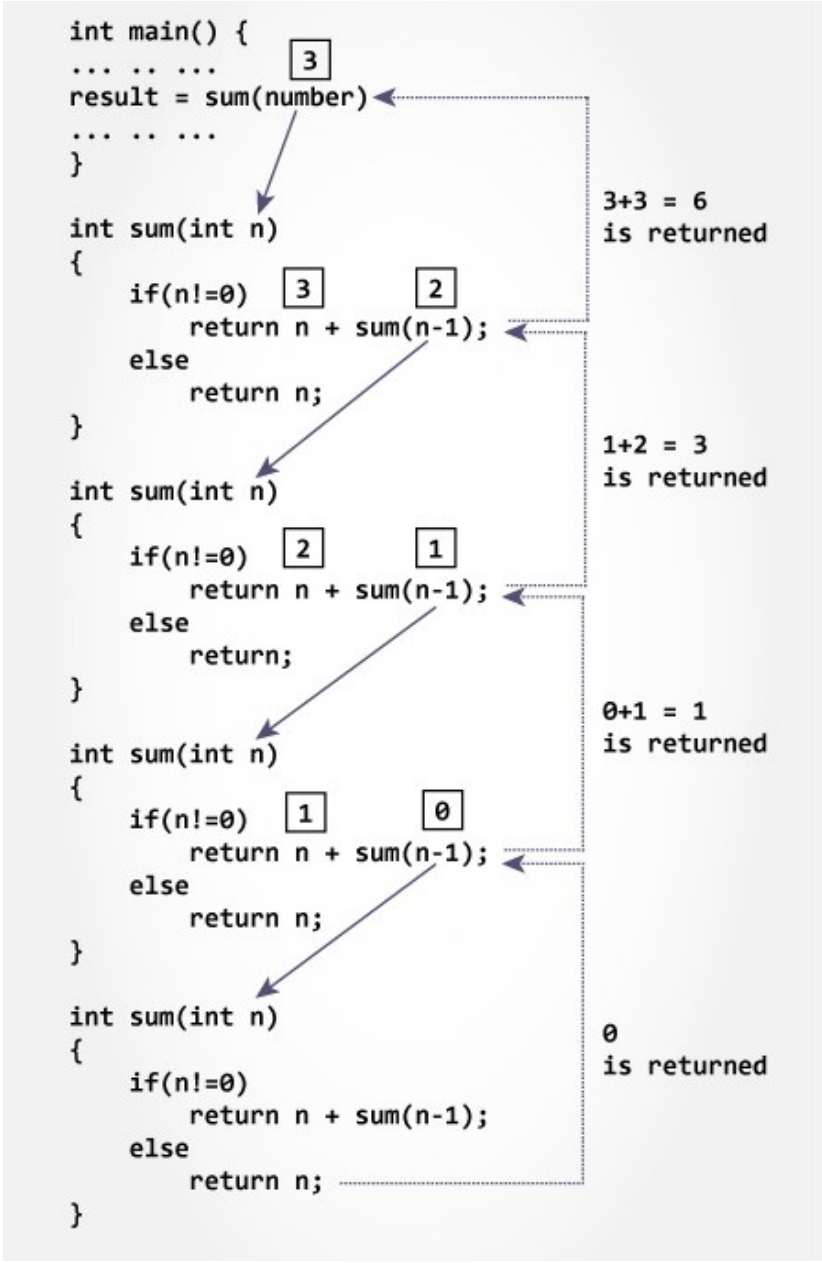
Output

```
Enter a positive integer:3
sum = 6
```

- Initially, the sum() is called from the main() function with number passed as an argument.

- Suppose, the value of num is 3 initially. During next function call, 2 is passed to the sum()function. This process continues until num is equal to 0.

- When num is equal to 0, the if condition fails and the else part is executed returning the sum of integers to the main() function.



[1]

Advantages and Disadvantages of Recursion

- Recursion makes program elegant and more readable.
- However, if performance is vital then, use loops instead as recursion is usually much slower.
- Note that, every recursion can be modeled into a loop.
- **Recursion Vs Iteration?**
  - Need performance, use loops, however, code might look ugly and hard to read sometimes.
  - Need more elegant and readable code, use recursion, however, you are sacrificing some performance.

Advantages and Disadvantages of Recursion

[2]

- So, which method is best?
- Well, obviously it depends on what you're trying to do - not all processes will provide an obvious recursive solution.
- Where they exist, recursive solutions tend to be more elegant, but can potentially be more difficult to understand (an important point to bear in mind if other people are going to be maintaining your code).
- Recursive functions may also be executed more slowly (depending on your environment).
  - Each time a function is called, certain values are placed onto the stack - this not only takes time, but can eat away at your resources if your function calls itself many times.
- In extreme examples you could run out of stack space. Functions that just iterate make no such demands on stack space, and may be more efficient where memory is limited.

Rekürsif Fonksiyonlar

- Rekürsif olarak faktöriyel hesabını yapan C kodunu yazınız.
- İki sayının en büyük ortak bölenini rekürsif olarak bulan C kodunu yazınız.
- Bir diziyi rekürsif fonksiyon kullanarak tersine çeviren C kodunu yazınız.
- İki sayının en küçük ortak katını rekürsif olarak bulan C kodunu yazınız.

Hazır Fonksiyonlar
Standard Library Functions

[3]

- C Standard library functions or simply C Library functions are inbuilt functions in C programming to handle tasks such as
  - Mathematical computations,
  - I/O processing,
  - String handling etc.
- The prototype and data definitions of the functions are present in their respective header files, and must be included in your program to access them.
- These functions are defined in the header file. When you include the header file, these functions are available for use.
- For example:
  - The printf() is a standard library function to send formatted output to the screen (display output on the screen). This function is defined in "stdio.h" header file.
- There is at least one function in any C program, i.e., the main() function (which is also a library function). This function is automatically called when your program starts.

Advantages of using C library functions [3]

- There are many library functions available in C programming to help you write a good and efficient program. But, why should you use it?

- Below are the 4 most important advantages of using standary library functions.

1. They work

- One of the most important reasons you should use library functions is simply because they work.
- These functions have gone through multiple rigorous testing and are easy to use.

2. The functions are optimized for performance

- Since, the functions are "standard library" functions, a dedicated group of developers constantly make them better.
- In the process, they are able to create the most efficient code optimized for maximum performance.

3. It saves considerable development time

- Since the general functions like printing to a screen, calculating the square root, and many more are already written. You shouldn't worry about creating them once again.
- It saves valuable time and your code may not always be the most efficient.

4. The functions are portable

- With ever changing real world needs, your application is expected to work every time, everywhere.
- And, these library functions help you in that they do the same thing on every computer.
- This saves time, effort and makes your program portable.

Library Functions – math.h

Fonksiyon Bildirimi	Açıklama	Örnek	Sonuç
`int abs(int x);`	x tamsayısının mutlak değerini hesaplar	`abs(-4)`	4
`double fabs(double x);`	x gerçel sayısının mutlak değerini hesaplar	`fabs(-4.0)`	4.000000
`int floor(double x);`	x'e (x'den büyük olmayan) en yakın tamsayıyı gönderir	`floor(-2.7)`	-3
`int ceil(double x);`	x'e (x'den küçük olmayan) en yakın tamsayıyı gönderir	`ceil(-2.7)`	-2
`double sqrt(double x);`	pozitif x sayısının karekökünü hesaplar	`sqrt(4.0)`	2.000000
`double pow(double x, double y);`	x^y değerini hesaplar	`pow(2., 3.)`	8.000000
`double log(double x);`	pozitif x sayısının doğal logaritmasını hesaplar, ln(x)	`log(4.0)`	1.386294
`double log10(double x);`	pozitif x sayısının 10 tabanındaki logaritmasını hesaplar	`log10(4.0)`	0.602060
`double sin(double x);`	radyan cinsinden girilien x sayısının sinüs değerini hesaplar	`sin(3.14)`	0.001593
`double cos(double x);`	radyan cinsinden girilien x sayısının kosinüs değerini hesaplar	`cos(3.14)`	-0.999999
`double tan(double x);`	radyan cinsinden girilien x sayısının tanjant değerini hesaplar	`tan(3.14)`	-0.001593
`double asin(double x);`	sinüs değeri x olan açıyı gönderir. Açı -pi/2 ile pi/2 arasındadır	`asin(0.5)`	0.523599
`double acos(double x);`	cosinüs değeri x olan açıyı gönderir. Açı -pi/2 ile pi/2 arasındadır	`acos(0.5)`	1.047198
`double atan(double x);`	tanjant değeri x olan açıyı gönderir. Açı -pi/2 ile pi/2 arasındadır	`atan(0.5)`	0.463648

Library Functions – stdlib.h

Fonksiyon Bildirimi	Açıklama	Örnek	Sonuç
`int atoi(const char *s);`	Bir karakter topluluğunu tamsayıya çevirir	`atoi("-12345")`	-12345
`long atol(const char *s);`	Bir karakter topluluğunu uzun tamsayıya çevirir	`atol("1234567890")`	1234567890
`double atof(const char *s);`	Bir karakter topluluğunu gercel sayıya çevirir	`atof("-123.546")`	-123.456
`void exit(int durum);`	Programı sonlandırarak kontrolü işletim sistemine geri verir.	`exit(1)`	-
`int rand(void);`	0 ile `RAND_MAX` arasında rastgele sayı üretir. `RAND_MAX`, `stdlib.h` içinde tanımlanmış bir sembolik sabittir	`rand()`	50485132
`max(a,b)`	`stdlib.h`'de tanımlanmış iki sayıdan en büyüğünü bulan makro fonksiyon	`max(5, 9)`	9
`min(a,b)`	`stdlib.h`'de tanımlanmış iki sayıdan en küçüğünü bulan makro fonksiyon	`min(5, 9)`	5

Library Functions – Ctype.h

Fonksiyon Bildirimi	Açıklama	Örnek	Sonuç
`isalpha(c)`	c bir harf ise 0 dan farklı, değilse 0 gönderir	`isalpha('a')`	1
`isalnum(c)`	c A-Z, a-z veya 0-9 arasında ise 0 dan farklı, değilse 0 gönderir	`isalnum('a')`	1
`isascii(c)`	c bir ASCII karakter ise 0 dan farklı, değilse 0 gönderir	`isascii('a')`	1
`isdigit(c)`	c bir rakam ise 0 dan farklı, değilse 0 gönderir	`isdigit('4')`	2
`islower(c)`	c a-z arasında ise 0 dan farklı, değilse 0 gönderir	`islower('P')`	0
`isupper(c)`	c A-Z arasında ise 0 dan farklı, değilse 0 gönderir	`islower('P')`	4
`toascii(c)`	c sayısı ile verilen ASCII koda sahip karakteri elde eden makro	`toascii(65)`	A
`tolower(c)`	c karakterini küçük harfe çevirir	`tolower('D')`	d
`toupper(c)`	c karakterini büyük harfe çevirir	`toupper('b')`	B

Library Functions – string.h

Fonksiyon Bildirimi	Açıklama
int strlen(str)	str nin kaç karakterden oluştuğunu hesaplar
char * strcat(char *s1, const chat *s2)	s1'in sonuna s2'yi kopyalar.
char *strchr(const char *str, int c)	Str içinde c karakterinin arar. Sonuc olarak c karakterinden sonra gelen metni döndürür.
int strcmp(const char *str1, const char *str2)	İki stringi karşılaştırır.
char *strcpy(char *dest, const char *src)	Bir stringe başka bir stringi kopyalar.
char *strncpy(char *dest, const char *src, size_t n)	Dest içerisine srcu n karakter kopyalar.
char *strstr(const char *haystack, const char *needle)	Samanlıkta iğne arar.
char *strtok(char *str, const char *delim)	Belirli bir stringe göre string içinde stringleri parse eder.

Hazır Fonksiyonlar

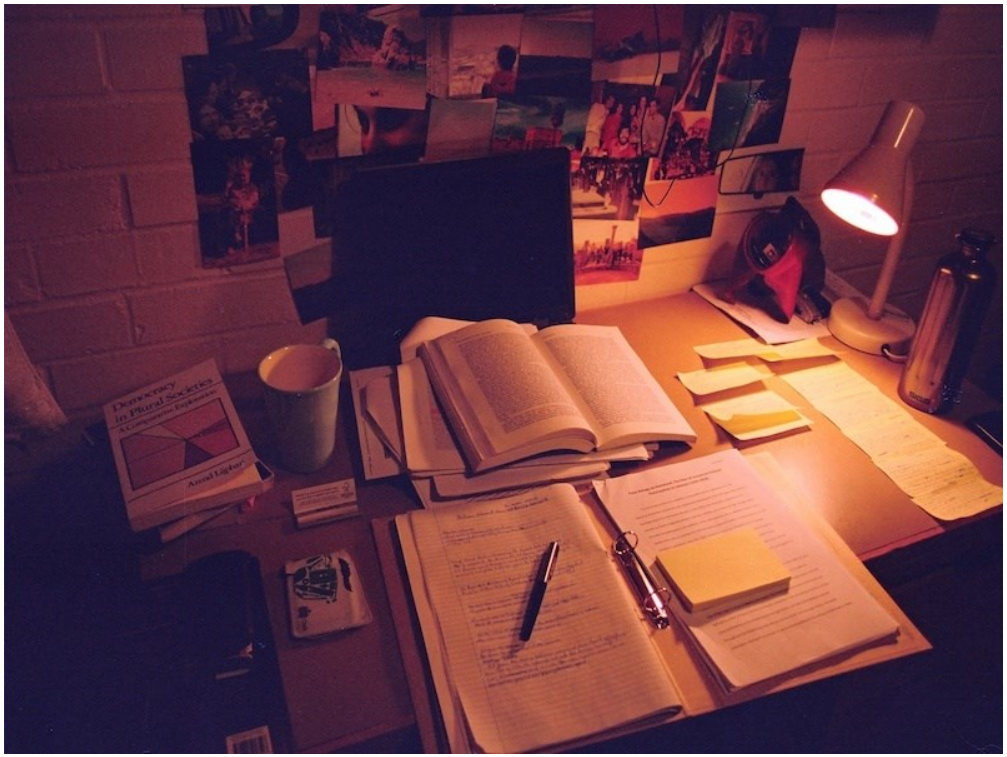
- Bir dizi içinde birbirinden farklı 1-100 arasında 10 tane sayıyı rasgele olarak tutan C kodunu yazınız.

- Bir stringteki büyük harfi küçük harfe, küçük harfi büyük harfe çeviren C kodunu yazınız.

- strcpy, strncpy, strcmp

Önümüzdeki Hafta

- VİZE SINAVI



Kaynakça

1. https://www.programiz.com/c-programming/c-recursion
2. https://www.advanced-ict.info/programming/recursion.html
3. https://www.programiz.com/c-programming/library-function