



++ Eğitimi



Akademi: **DEPAR** AKADEMİ

<https://www.deparakademi.com.tr/>

Eğitmen: **Bülent Çobanoğlu** (Bülend Hoca)

++ Eğitimi-5

- –: Çoklu istisna fırlatma-yakalama
- –: Kendi Exception (istisna) sınıfımızı nasıl yazarız?
- –: Pointers to Pointer
- –: Referans & Pointer Farkı
- –: Önişlemci Komutları
- –: Dizilerde Sıralama - Arama
- –: Dosyalama

Çoklu İstisna fırlatma ve yakalama

```
string fruits[] = {"banana", "melon", "cherry", "apple", "kiwi", "grape", "mango"};
```

- Kullanıcıdan aldığı indis numarasına göre “Benim favori meyvem..: ...” şeklinde istedeki bir meyveyi yazdıran programı kodlayalım.
- Program, negatif indis ya da indis taşması durumunda aşağıdaki gibi hata mesajlarını vermeli ta ki doğru indis numarası girilene kadar...

```
indis no..: -6  
Exception: negatif indis -  
indis no..: 99  
Exception: indis taşması 99  
indis no..: 2  
Benim favori meyvem..:cherry
```

Çoklu İstisna fırlatma

```
#include<iostream>
using namespace std;
int main () {
    string fruits[] = {"banana", "melon", "cherry", "apple", "kiwi", "grape", "mango"};
    int indis;
    while(1) { //sonsuz döngü
        try {
            cout<< "indis no..: ";
            cin>> indis;
            if (indis > 6 ) throw indis;
            if (indis < 0) throw '-';
            cout<<"Benim favori meyvem..:"<< fruits[indis]<<endl;
            break;
        }
        catch (int i) {
            cout << "Exception: ";
            cerr << "indis taşması " << i << endl;
        }
        catch (char st) {
            cout << "Exception: " ;
            cerr << "negatif indis " << st << endl;
        }
    }
    return 0; }
```

Kendi istisna sınıfını yazma

benim_ex.h × div_zero.cpp isSimetrik.cpp benim_try.cpp

home > bull > C benim_ex.h > ...

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  class benim_Exception {
6  |   public:
7  |   benim_Exception(const string msg) {
8  |       cerr << "Hatasız kul olmaz! " << msg << endl;
9  |   }
10 };
```

Kendi istisna sınıfımı yazma

```
#include<iostream>
#include"benim_ex.h"
using namespace std;
int main() {
    double a,b;

    try {
        cout<<"iki sayı gir..: ";
        cin>> a >> b;
        if (b==0) throw benim_Exception("sıfıra bölme hatası");
        cout<<"Sonuç..:"<<(a/b)<< endl;

    } catch (benim_Exception &ex) {"\n";}
    return 0;
}
```

Parametresi Dizi olan Fonksiyonlar

```
#include <iostream>
using namespace std;
const int N = 2;
//Fonksiyon parametresi: V dizisi
void f1(int V[]) {
    V[0] = 2;
}
//Fonksiyon parametresi: M matrisi
void f2(int M[N][N]) {
    M[0][0] = 4;
}
//Ana program
int main() {
    int A[N][N];
    f1(A[1]); // A[1][0]= 2
    f2(A);    // A[0][0]= 4
    cout << A[0][0] << "\t" << A[0][1] << "\n";
    cout << A[1][0] << "\t" << A[1][1] << "\n";
    return 0;
}
```

Pointers to Pointer

- Pointer de birer nesne olduğuna ve bellekte yer kapladığına göre onların da adresleri alınabilir.
- T türünden bir pointerin adresi T* türünden bir pointer'e yerleştirilebilir. Bu durumda iki adet ** atomu kullanılır. Örneğin:

- `int **ppi;`

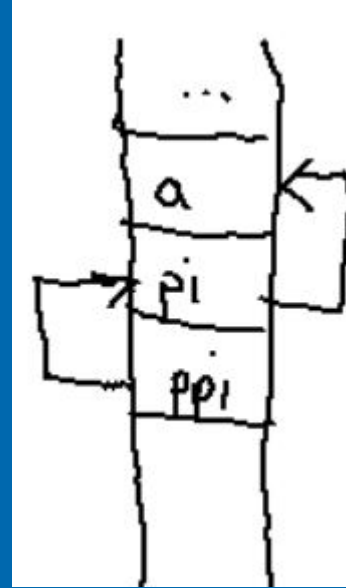
```
int a = 10;  
int *pi;  
int **ppi;
```

```
pi = &a;  
ppi = &pi;
```


Pointers to Pointer

```
int a = 10;  
int *pi;  
int **ppi;
```

```
pi = &a;  
ppi = &pi;
```



Pointers to Pointer ve Diziler

Bir pointer dizini (string dizi) bir fonksiyona geçirmek istesek fonksiyonun parametre değişkeni pointer'ı gösteren pointer olmalıdır

```
#include <stdio.h>

//fonksiyon prototipi
void show_names(char **ppnames);

//main fonksiyon
int main() {
    char *names[] = {"ali", "veli", "hakan", "ayse", "fatma", NULL };
    show_names(names);
    return 0;
}

//show names fonksiyonu
void show_names(char **ppnames) {
    int i;
    for (i = 0; ppnames[i] != NULL; ++i)
        printf("%s\n", ppnames[i]);
}
```

Pointers to Pointer ve Diziler

Burada

```
void show_names(char **ppnames)
{
    ...
}

ile,
```

```
void show_names(char *ppnames[])
{
    ...
}
```

eşdeğerdir.

String Dizilerde Pointer

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
void show_names(string *names, int size){
```

```
    for (int i = 0; i < size; i++)
```

```
        cout << names[i] << '\n';
```

```
}
```

```
int main() {
```

```
    string names[] = {"ali", "veli", "hakan", "aysem", "fatma"};
```

```
    show_names(names, sizeof names/sizeof names[0]);
```

```
}
```

C++ Referanslar

- ❑ Referanslar adres temsilcisidir
- ❑ Referanslar işaretçilerin görevlerini kısmi olarak yapabilirler
- ❑ Referanslar değişken değildir. Değişken değeri atanır.
- ❑ Bellekte yer tutmaları mümkün değildir
- ❑ & karakteri ile tanımlanabilirler
- ❑ Tanımlama anında atamaları da yapılmalıdır.
- ❑ Referanslar C' de mevcut değildir
- ❑ Kullanılmaları çok pratiktir.

Referans Tanımı-Ataması

```
int a = 3;  
int &refA = a; // Başlangıç ataması şart!!
```

```
cout << a << endl;    // 3  
cout << refA << endl; // 3  
cout << &a << endl;   // 0x 7fffffffddac  
cout << &refA << endl; // 0x 7fffffffddac
```

```
refA = 5;                // refA sonradan değeri değişebilir  
cout << a << endl;      // 5  
cout << refA << endl;  // 5
```

Pointer - Referans Farkı

```
#include<iostream>

using namespace std;

void change(int *ptr1,int *ptr2){

    (*ptr1)++;

    (*ptr2)--;

}

int main( )

{

    int a=10,b=20;

    cout<<"a="<<a<<"\n";

    cout<<"b="<<b<<"\n";

    change (&a, &b);

    cout<<"a="<<a<<"\n";

    cout<<"b="<<b<<"\n";

    return 0;

}
```

Çıktısı nedir?

Pointer - Referans Farkı

```
#include<iostream>
using namespace std;
void change(int *ptr1,int *ptr2){
    (*ptr1)++;
    (*ptr2)--;
}
int main( ) {
    int a=10,b=20;
    cout<<"a="<<a<<"\n";
    cout<<"b="<<b<<"\n";
    change (&a, &b);
    cout<<"a="<<a<<"\n";
    cout<<"b="<<b<<"\n";
    return 0;
}
```

Çıktısı

```
a=10
b=20
a=11
b=19
```


Pointer - Referans Farkı

```
#include<iostream>
using namespace std;
```

```
void change(int &ref1,int &ref2){
    ref1++;
    ref2--;
}
```

```
int main( ) {
    int a=10,b=20;
    cout<<"a="<<a<<"\n";
    cout<<"b="<<b<<"\n";
    change(a,b);
    cout<<"a="<<a<<"\n";
    cout<<"b="<<b<<"\n";
    return 0;
}
```

Çıktısı

```
a=10
b=20
a=11
b=19
```

Pointer - Referans Farkı

```
void change(int &ref1,int &ref2){  
    ref1++;  
    ref2--;  
}
```

```
//Nasıl çağrılır?  
change(a,b);
```

```
void change(int *ptr1,int *ptr2){  
    (*ptr1)++;  
    (*ptr2)--;  
}
```

```
//Nasıl çağrılır?  
change(&a,&b);
```

Dizilerde Sıralama -bubbleSort

Kabarcık Sıralama Algoritması:

```
for (i = 0; i < n-1; i++) {  
    for (j = i+1; j < n; j++) {  
        if (A[i] > A[j])  
        { //Yer değiştirme işlemi  
            Bos=A[i];  
            A[i]=A[j];  
            A[j]=Bos;  
        }  
    }  
}
```

Tüm sayılar küçükten büyüğe sıralanana kadar **TEKRARLA**

1. Sırasıyla yan yana (*komşu*) iki sayıyı karşılaştır.
2. Eğer soldaki sayı sağdakinden büyükse yer değiştir.
3. Bir sonraki sayıdan karşılaştırmaya devam et.

Dizilerde Sıralama - insertion

Araya eklemeli sıralama (*insertion sort*) algoritmasında; sıralanacak sayılar iki kısma ayrılır;

- a) *Sıralanmış* kısım ve
- b) *Sıralanmamış* kısım olmak üzere.

Sıralanmamış kısımdan elemanlar alınarak sıralı kısımda uygun yerde araya yerleştirilir. Dolayısıyla sola doğru bir kaydırma işlemi söz konusudur.

Algoritması:

```
for (i=1; i<n; i++)  
{  
    ekle = A[i];  
    while (i > 0 && (A[i-1] < ekle))  
    {  
        A[i] = A[i - 1];  
        i = i - 1;  
    }  
    A[i] = ekle;  
}
```

İlk sıradaki sayı ($A[0]$) başlangıçta sıralanmış kabul edilir.

Tüm sayılar büyükten küçüğe sıralanana kadar **TEKRARLA**

1. Sıralanmamış ilk sayıyı al ($A[i]$);
2. Bu sıralanmamış sayıyı ($A[i]$), solundaki sıralanmış sayı ($A[i-1]$) ile karşılaştır. Eğer soldaki sayı sağdakinden küçükse yer değiştir.

(Bu işleme (tüm sayılar için); kendisinden önceki sayının değerini daha büyük bulana kadar devam edilir.)

Dizilerde Sıralama - Selection

Seçmeli sıralama (*selection sort*) algoritmasında; dizinin ilk ya da son elemanı seçilir. Sonra dizi içerisindeki en küçük eleman aranır ve bu eleman dizinin **ilk/son** elemanı ile yer değiştirilir. Daha sonra ikinci en küçük eleman aranır ve bu eleman dizinin ikinci (**ilk+1/son-1**) elemanı ile yer değiştirilir. Bu işlem, dizinin son elemanına kadar tekrarlanarak dizi sıralanır.

Seçmeli Sıralama Algoritması:

```
for (i=0; i<n-1; i++) {  
    int enk = i; // ilk eleman en küçük seçildi  
    for (j=i+1; j<n; j++) {  
        if (a[enk] > a[j]) {  
            enk = j; } //Yeni en küçük eleman  
    }  
    //Yer değiştirme işlemi  
    Bos = a[i];  
    a[i] = a[enk];  
    a[enk] = Bos;  
}
```

Dizilerde Sıralama- TASK

```
#define MAX 10
main()
{
    int a[MAX]={2, 6, 4, 8, 10, 12, 89, 68, 45, 37};
    int b[MAX]={12, 16, 24, 38, 50, 112, 29, 18, 435, 137};

    printf("a dizisinin sırasız hali\n");
    diziYaz(a,MAX);
    bubblesort(a,MAX);
    printf("a dizisinin bubblesort ile sıralı hali\n");
    diziYaz(a,MAX);

    printf("b dizisinin sırasız hali\n ");
    diziYaz(b,MAX);
    insertionsort(b,MAX);
    printf(" b dizisinin insertionsort ile sıralı hali\n ");
    diziYaz(b,MAX);
}
```

Bülend Hoca (Dr. Bülent
Çobanoğlu)

Dizilerde Arama

Linear Arama (linear search):

```
int linearArama(int a[ ], int aranan, int max)
{
    int i;
    for (i=0;i<=max-1;i++)
        if (a[i]==aranan)
            return i;
    return -1;
}
```

İkili Arama (binary search):

```
int ikilikArama(int a[], int aranan, int max)
{
    int ilk,son,orta;
    ilk=0;
    son=max-1;
    orta=(ilk+son)/2;
    while (ilk<=son)
    {
        if (a[orta]>aranan)
            son=orta-1;
        else if (a[orta]<aranan)
            ilk=orta+1;
        else
            return orta;
        orta=(ilk+son)/2;
    }
    return -1;
}
```

Dizilerde Arama

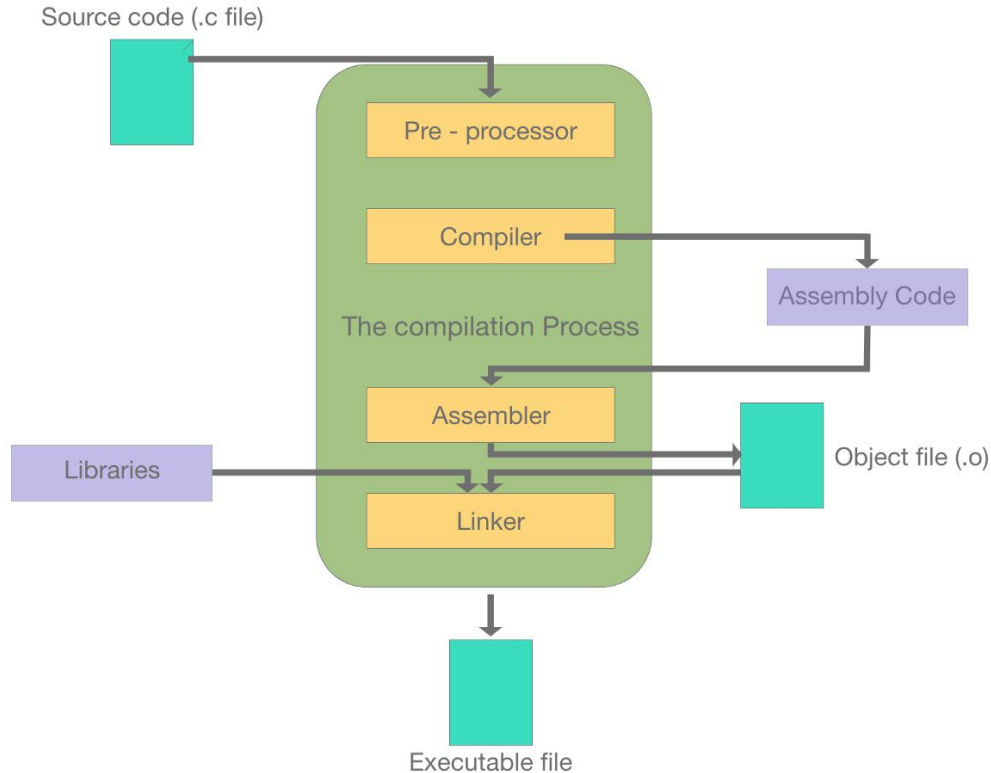
TASK:

Bir karakter dizisi içerisinde arama yapan fonksiyonu işaretçi kullanarak kodlayınız. Arama fonksiyonunda aranan karakter bulunursa karakterin bulunduğu yerin adresi bulunamazsa 0 (NULL) değeri geri döndürülecektir.

Dizilerde Arama

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  //Arama fonksiyonu
5  char *araFonk(char *st, int kr){
6  while (*st != '\0'){
7      if (*st == kr)
8          return st;
9      st++;
10 }
11 if (kr == '\0') return st;
12 else return NULL; /*hiçbir koşul sağlanamazsa NULL döndürür */
13
14 }
15 //Ana program
16 int main() {
17     char ata[] = "ATATURK";
18     char *pr;
19     pr = araFonk(ata, 'K');
20     if (pr)
21         cout<<" Adresi: " << &pr << endl;
22     else
23         cout<<"Aranan Bulunamadı";
24     return 0;
25 }
```

Önişlemci Kavramı (pre-processor)



Önişlemci Komutları

C/C++ '#' karakteri ile başlayan satırlar ön işlemciye ilişkindir. Önişlemci komutu ön işlemciye ne yapması gerektiğini anlatır. 10 dan fazla önişlemci komutu vardır. Ancak en çok kullanılanı **#include** ve **#define** komutlarıdır.

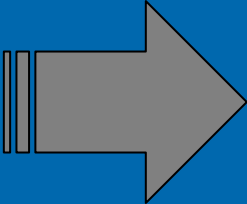
```
#include <stdio.h>
```

```
#include "a.c" //aynı satırda tek #include olabilir...
```

```
#define SIZE 100
```

```
#define MAX(a, b) ((a) > (b) ? (a) : (b)) //makro
```

Diğer Önışlemci Komutları: #if, #else, #elif ve #endif

<pre>#if MAX > 10 #else ... #endif</pre>	<pre>#if MAX > 10 #else #if MAX>50 ... #endif #endif</pre>		<pre>#if MAX > 10 #elif MAX>50 ... #endif #endif</pre>
--	---	---	--

Diğer Önışlemci Komutları: #ifdef, #else ve #endif

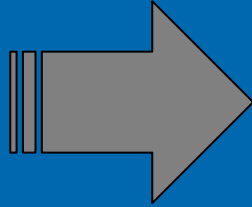
#ifdef komutunu

#define ile

tanımlanmış

sembolik sabit

izlemelidir.



```
#include <stdio.h>
```

```
#define TEST
```

```
int main(void) {
```

```
    #ifdef TEST
```

```
        printf("TEST define edilmiş!\n");
```

```
    #else
```

```
        printf("TEST define edilmemiş!\n");
```

```
    #endif
```

```
    return 0;}
```

```
#include <iostream>
#include <time.h>
using namespace std;
#ifdef __WIN32
    #include <Windows.h>
#else
    #include <unistd.h>
#endif

void sleep(int milliseconds) // Cross-platform sleep function
{
    #ifdef __WIN32
        Sleep(milliseconds);
    #else
        usleep(milliseconds * 1000);
    #endif // __WIN32
}

int main()
{
    int s=0;
    cout << "Bu program 3 sn de bir ekrana Hi! yazar! :)" << endl;
    do {
        s++;
        printf("%d.",s);
        sleep(3000); //3000 ms
        cout << "Hi!  :)" << endl;
    }while (s<10);
}
```

Diğer Önışlemci Komutları: #ifndef, #else ve #endif

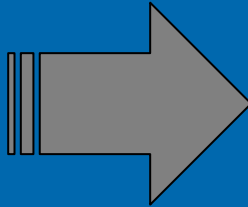
#ifndef komutu

#ifdef komutunun tam tersidir. Yani sembolik sabit tanımlanmamışsa #else kısmına kadarki bölüm, tanımlanmışsa #else ile #ifndef arasındaki kısım derlenir. Bu, aynı başlık dosyasının, sembolik sabitin birden fazla koda dahil edilmesini önler.

```
#ifndef TEST
```

```
#define TEST 10
```

```
#endif
```



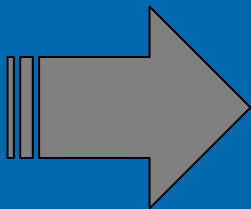
Burada TEST sabiti daha önce tanımlanmamışsa tanımlaması yapılır. Daha önce tanımlandı ise tekrar tanımlanmaz...

Genel Önışlemci Komutları:

#undef

#undef komutu

#define ile tanımlanan
sembolik sabiti kaldırır.



```
#define TEST 10
```

```
...
```

```
#define TEST 100 //hata
```

```
#undef TEST //önce kaldır
```

```
#define TEST 100
```

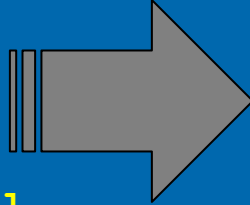
```
//sonra değeri güncelle
```


Genel Önışlemci Komutları:

#error

#error komutu önışlemci tarafından derleme işlemini fatal error ile sonlandırır. Komutun yanında bir yazı bulunur.

Bu yazı ekrana hata mesajı olarak yazdırılır (Yazının iki tırnak içerisinde olması gerekmez.)



```
#include <stdio.h>
#ifdef LINUX
#error "Bu program linux de derlenmez"
#endif
int main(void)
{
    return 0;
}
```

Genel Önışlemci Komutları:

#pragma

#pragma anahtar sözcüğünü başka bir komut sözcüğü izler. Bu komut sözcüğü standart değildir. Yani #pragma komutu standarttır ancak onun yanındaki komut sözcüğü derleyiciden derleyiciye değişebilmektedir. Her derleyicinin pragma komutları diğerinden farklı olabilir. Pragma komutları belli bir derleyiciye özgü işlemlerin yaptırılması için kullanılır



```
#include <stdio.h>
#pragma inline
int gcd( int a, int b ) {
    int result ;
    /* Compute Greatest Common Divisor using Euclid's Algorithm */
    asm volatile ( "movl %1, %%eax;"
                   "movl %2, %%ebx;"
                   "CONTD: cmpl $0, %%ebx;"
                   "je DONE;"
                   "xorl %%edx, %%edx;"
                   "idivl %%ebx;"
                   "movl %%ebx, %%eax;"
                   "movl %%edx, %%ebx;"
                   "jmp CONTD;"
                   "DONE: movl %%eax, %0;" : "=g" (result) : "g" (a), "g" (b)
    );
    return result ;
}

int main() {
    int first, second ;
    printf( "Enter two integers : " ) ;
    scanf( "%d%d", &first, &second );
    printf( "GCD of %d & %d is %d\n", first, second, gcd(first, second) ) ;
    return 0 ;
}
```

Dosyalama

İşletim sistemi tarafından organize edilen ikincil belleklerde tanımlanmış bölgelere dosya (file) denilmektedir.

Bir dosyanın yerini belirten yazısal ifadelere yol ifadeleri (**path**) denilmektedir. Yol ifadeleri mutlak (absolute) ve görelî (relative) olmak üzere ikiye ayrılır. Yol ifadesinin ilk karakteri Windows'ta '\', UNIX/Linux'ta '/' ise böyle yol ifadelerine mutlak (absolute) yol ifadeleri denilmektedir.

`"/A/B/C.txt"` → linux, mac,...

`"C:\A\B\C.txt"` → windows (dikkat!) `"C:\\A\\B\\C.txt"`

Dosya İşlemleri

Programlama dillerine göre farklılık gösterse de dosyalarla ilgili yapılan işlemlerde genel olarak aşağıdaki adımlar takip edilir;

1. **Dosyanın açılması**
2. **Dosya ile ilgili işlem yapılması** (*kayıt okuma, yazma, düzeltme, silme gibi*)
3. **Dosyanın kapatılması**

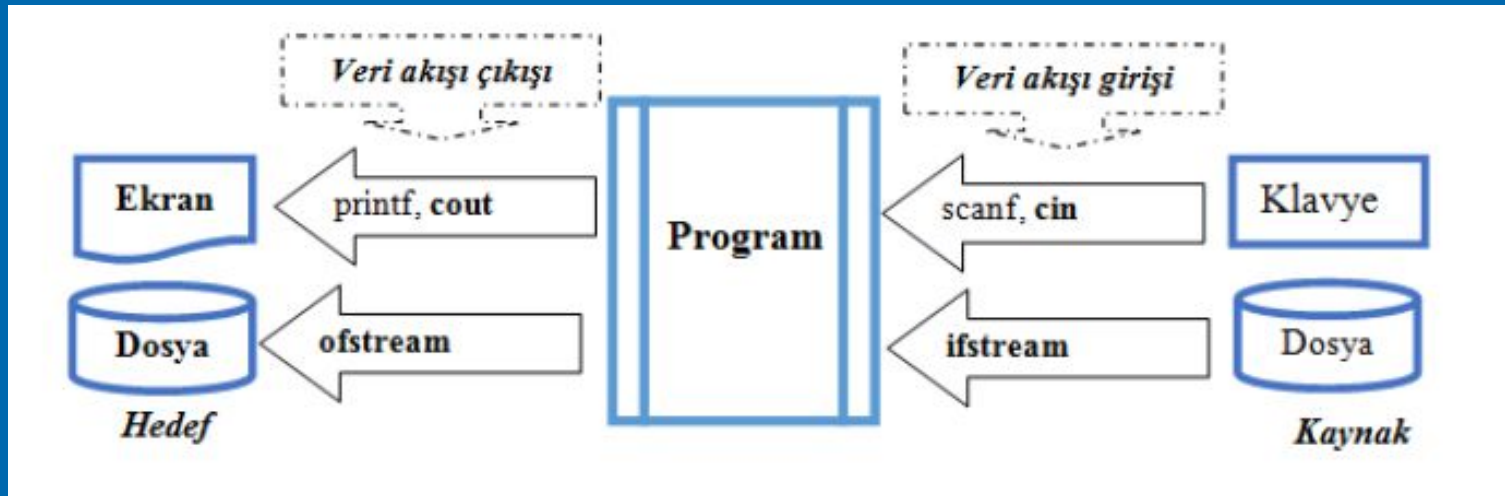
fopen ... fclose()

```
FILE * dosya = fopen(dosyaadi, "r"); // Dosya okuma amaçlı açıldı  
FILE * dosya = fopen(dosyaadi, "w"); // Dosya yazma amaçlı açıldı  
FILE * dosya = fopen(dosyaadi, "a"); // Dosya ekleme amaçlı açıldı
```

C/C++ dillerinde ortak olan dosya işlem fonksiyonları tablo 16.1'de verilmiştir.

Klasik dosya işlem fonksiyonları	Açıklama
fopen()	Dosya açma komutudur. Örneğin; "dosyam.txt" adlı bir dosyayı "r (read-okuma)" modunda açarak adresini fp isimli değişkende saklamak için; fp=fopen("dosyam.dat","r") ;
fclose()	Dosyayı kapatır. Kullanım şekli; fclose(fp) ;
fscanf/fgets()	Dosyadan veri okur. Kullanım şekli; fgets(str, 12, fp); fscanf(fp, "%s", &str);
fprintf/fputs()	Dosyaya veri yazar. Kullanım şekli; fputs("Ali", fp); fprintf(fp, "Ali");

C++ Dosya İşlemleri



Metin Dosyasına Veri Yazmak

Dosyaya veri yazma işlem algoritmasını C/C++ dillerine göre aşağıdaki gibi ifade edebiliriz;

C++ dosya işlem algoritması	C/C++ ortak dosya işlem algoritması
<pre>#include <fstream> ofstream dosya (dosya_adı ve yolu); //... //... dosya ile ilgili işlemler; //..... dosya.close();// Dosyayı kapat</pre>	<pre>FILE *dosya; // dosya göstericisi dosya = fopen(const char dosya_adı, const char açma_amacı); //... //dosya ile ilgili işlemler; //... fclose(dosya); // Dosyayı kapat</pre>

Metin Dosyasına Veri Yazmak

Dosyaya veri yazma işlem algoritmasını C/C++ dillerine göre aşağıdaki gibi ifade edebiliriz;

C Dili Kodlaması	C++ Dili Kodlaması
<pre>#include <stdio.h> #include <stdlib.h> #include <string.h> int main() { FILE *dY; char str[100]; dY=fopen("C:\\Test\\MetinC. txt","w"); printf ("Metni giriniz\n"); fflush(stdout); gets(str); fprintf(dY, "%s\n", str); fclose(dY); return 0; }</pre>	<pre>#include <iostream> #include <fstream> using namespace std; int main() { ofstream dY("C:\\Test\\MetinC. txt"); string str; cout << "Metni giriniz" << endl; cin >> str; dY << str << endl; dY.close(); return 0; }</pre>

Metin Dosyasından Veri Okumak

C/C++ dillerinde dosya sonuna gelinmediği sürece tüm kayıtları taraması için **eof()** veya **feof()** fonksiyonu kullanılır. eof/feof fonksiyonu dosya sonuna gelince **true** değerini döndürür

0	1	2	3	4	5	...	n-1	Dosya işaretçisi
						...		EOF
EOF: Dosya sonuna gelinip gelinmediğini kontrol eden bir dosya işaretçisidir.								

C Kullanımı

```
while (!feof(dOku))  
{
```

C++ Kullanımı

```
while (!dOku.eof())  
{
```

Metin Dosyasından Veri Okumak

C/C++ dillerinde dosya sonuna gelinmediği sürece tüm kayıtları taraması için **eof()** veya **feof()** fonksiyonu kullanılır. eof/feof fonksiyonu dosya sonuna gelince **true** değerini döndürür

C Dili Programlaması	C++ Dili Programlaması
<pre>#include <stdio.h> #include <stdlib.h> int main() { FILE *dOku; char str[100]; dOku=fopen("C:\\Test\\MetinC. txt","r"); while (!feof(dOku)) { fscanf(dOku, "%s", &str); printf("%s\n", str); } fclose(dOku); return 0; }</pre>	<pre>#include <fstream> #include <iostream> using namespace std; int main() { ifstream dOku ("C:\\Test\\ MetinC.txt"); string str; while (!dOku.eof()) { str=dOku.get(); if (dOku.eof()) break; cout << str; } dOku.close(); return 0; }</pre>

Metin Dosyasından Veri Okumak

Bir text dosyasından verileri satır satır okuyup ekrana yazdıran programı kodlayalım. Örnek dosyamız ve içeriği;

```
> bull > ≡ sozluk.txt  
Pazartesi    Monday  
Salı         Tuesday  
Carşamba    Wednesday  
Perşembe     Thursday  
Cuma         Friday  
Cumartesi    Saturday  
Pazar        Sunday
```

Metin Dosyasından Veri Okumak

```
#include <iostream>
#include <string.h>
#include <fstream>
using namespace std;

int main() {
    ifstream inFile;
    string tr, en;

    inFile.open("sozluk.txt");
    cout<<"Türkçe \t İngilizce\n";
    cout<<"-----\t -----\n";
    while(!inFile.eof()){
        getline(inFile, tr, '\t'); // default is '\n' (newline).
        getline(inFile, en, '\n');
        cout << tr << "\t";
        cout << en << endl;
    }
    inFile.close();
    return 0;
}
```

Programın Çıktısı;

Türkçe	İngilizce
-----	-----
Pazartesi	Monday
Salı	Tuesday
Çarşamba	Wednesday
Perşembe	Thursday
Cuma	Friday
Cumartesi	Saturday
Pazar	Sunday

Metin Dosyasından Veri Okumak

TASK-0:

mevlana.txt dosyasını oluşturup, o dosyadan verileri okuyan programı kodlayalım...

[mevlana.txt dosyası içeriği](#)

I want to sing
Like the birds sing,
Not worrying about
Who hears or
What they think.



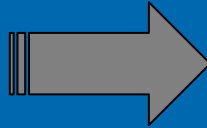
Metin Dosyasından Veri Okumak

TASK-1:

Oluşturulan mevlana.txt dosyasının içeriği birebir rumi.txt dosyasına kopyalayan bir programı kodlayalım...

mevlana.txt dosyası içeriği

I want to sing
Like the birds sing,
Not worrying about
Who hears or
What they think.



rumi.txt

I want to sing
Like the birds sing,
Not worrying about
Who hears or
What they think.



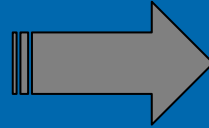
Metin Dosyasından Veri Ekleme

TASK-2:

Mevlana.txt dosyasının içeriği yeni kayıt ekleyen programı kodlayalım...

eklenecek_metin

Raise your words,
not voice.
It is rain that grows flowers,
not thunder.



mevlana.txt

I want to sing
Like the birds sing,
Not worrying about
Who hears or
What they think.

Raise your words,
not voice.
It is rain that grows flowers, not
thunder.



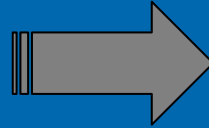
Metin Dosyasından Veri Eklemek

TASK-2:

Mevlana.txt dosyasının içeriği yeni kayıt ekleyen programı kodlayalım...

eklenecek metin

Raise your words,
not voice.
It is rain that grows flowers,
not thunder.



mevlana.txt

I want to sing
Like the birds sing,
Not worrying about
Who hears or
What they think.

Raise your words,
not voice.
It is rain that grows flowers, not
thunder.

Bir metin dosyasına veri/kayıt eklemek için dosya ekleme modunda açılmalıdır.
Programlama dillerine göre bu işlem;

- C++ için; `ofstream d1 ("dosya.txt", ios::app);`
- C için; `d1 = fopen("dosya.txt", "a");`

Şeklinde tanımlanır.

THE END

Teşekkürler

e-mail:

cobanoglubulent@gmail.com

