AKIŞLAR

Dosya ve Akış Nedir?

Yazdığımız programlarda sadece girdi ve çıktı işlemleri yaparsak, program çalıştığı sürece veriler var olur, program sonlandırıldığında o verileri tekrar kullanamayız. Bunun için elektrik kesildiğinde bellekteki veriler kaybolacağından harici hafıza ortamında (external memory) verileri saklamak gerekir.

Kısaca dosya (file), verilerin bir araya geldiği (collection of data) ikincil saklama ortamıdır (seconder storage). Bu ikincil ortam; Bir bilgisayar belleği (memory) olabileceği gibi, verilerin elektrikler kesildiğinde kaybolmayacağı sabit disk (hard disc), ya da bir başka ortama/bilgisayara veri gönderen (send) veya alan (receive) modem ve benzeri bir cihaz (device) olabilir.

Veriler dosyalara aynı nehirde birbirinin peşi sıra giden tekneler gibi blok-blok ya da bayt-bayt gönderilir ya da dosyadan alınırlar. Bu nedenle dosyalara veri taşıyan ve veri getiren nesnelere nehir veya akış (stream) adı verilir. Dosyaya giden nehre hangi veriyi koyarsak onu nehrin sonunda onu dosyaya koyar. Benzer şekilde nehirden hangi sırada veri gelirse o sırada verileri nehirden alırız.

C++ dilinde giriş çıkış işlemleri akışlar aracılığıyla yapılır. Bu işlemler için soyut sınıflar şunlardır:

- Metin okumak için `std::istream`.
- Metin yazmak için `std::ostream`.
- Karakterleri okumak veya yazmak için `std::streambuf`.
- Biçimlendirilmiş girdi için `>>` işleci kullanır.
- Biçimlendirilmiş çıktı için `<<` işleci kullanır.
- Akışlar, örneğin biçimlendirmenin ayrıntıları ve harici kodlamalar ile dahili kodlama arasındaki çeviri için `std::locale` kullanır.

Şimdiye kadar, standart girdiden yani klavyeden okuma ve standart çıktıya yani konsola yazma için sırasıyla `cin` ve `cout` nesnelerini sağlayan akışların (stream) içinde tanımlı olduğu `iostream` başlığını kullandık.

Standart Akışlar

C++ dili programcıya, girdi ve çıktı işlemleri (input output operation) için hazır ve sürekli açık akış (stream) sunar. Üç metin akış nesnesi önceden `iostream` başlık dosyasında tanımlanmıştır;

Standart Akışlar	Akış	Açıklama
Standart Giriş	`cin`	Standart giriş akışıyla ilişkilendirilir, geleneksel klavye girişini okumak için kullanılır. Program başlangıcında, akış yalnızca ve yalnızca akışın etkileşimli bir cihaza başvurmadığı belirlenirse tamamen arabelleğe alınır.
Standart Çıktı	`cout`	Standart çıktı akışıyla ilişkilendirilir, geleneksel ekrana çıktıyı yazmak için kullanılır. Program başlangıcında, akış yalnızca ve yalnızca akışın etkileşimli bir cihaza başvurmadığı belirlenirse tamamen arabelleğe alınır.
Standart Hata	`cerr`	Standart hata akışıyla ilişkilendirilir, hata çıktısını kullanıcı ekranına yazmak için kullanılır. `cerr` verileri hemen çıktı olarak verir, yani verileri bir arabelleğe kaydetmez. Bu, hata mesajları için idealdir, çünkü bunların hemen görüntülenmesi gerekir.

Tablo 23. C++ Dilinde Standart Giriş Çıkış Akışları

Standart Almayan Akışlar

Ancak veriler üzerinde yapılan işlemleri kalıcı olarak diske saklamak veya kalıcı olan diskten verileri okumak için dosyalar kullanılır. Dosyalarla işlem yapmak için üç nesne kullanılır;

- `ofstream` sınıfı, verileri çıktıya taşıyan akışı (output stream) temsil eder ve dosyalar oluşturmak ve dosyalara bilgi yazmak için kullanılır.
- `ifstream` sınıf, verileri girdiden alan ve bize getiren akışı (input stream) temsil eder ve dosyalardan bilgi okumak için kullanılır.
- `fstream` sınıfı ise genel olarak hem veri gönderilen hem de veri alınan bir nehirdir. Hem `ofstream` hem de `ifstream` özelliklerine sahiptir; yani dosyalar oluşturabilir, dosyalara bilgi gönderebilir ve dosyalardan veri alabilir.

C++'da dosya işlemeyi gerçekleştirmek için, kaynak kodunuza `<iostream>` veya `<fstream>` başlık dosyalarını `#include` ön işlemci yönergesi (preprocessor direvtive) ile dahil edilmesi gerekir.

Bir dosyadan okuyabilmeniz veya dosyaya yazabilmeniz için önce dosyanın açılması gerekir. Bir dosyayı yazmak için açmak için `ofstream` veya `fstream` sınıflarından biri kullanılabilir. Yalnızca okuma amacıyla bir dosyayı açmak için `ifstream` nesnesi kullanılır. Aşağıda `fstream`, `ifstream` ve `ofstream` sınıflarının üyesi olan `open()` yöntemi ile dosyanın nasıl açılacağı gösterilmiştir;

```
void open(const char* dosya-adı, ios::openmode işlem-modu);
```

Akışın `open()` yönteminin ilk argümanı açılacak dosyanın adını ve konumunu belirtirken, ikinci argümanı ise dosyanın hangi modda açılacağını tanımlar. Dosya açma modlarından ikisini veya daha fazlasını bit düzeyi VEYA (bitwise or) işlemiyle birleştirebilirsiniz.

İşlem Modu	Açıklama
`ios::app`	Gönderilen bütün veri, dosyanın sonuna (append) eklenir.
`ios::ate`	Dosyanın sonunda (at the end) gidilerek hem okuma hem de yazma amacıyla dosya açılır.
`ios::in`	Dosya okumak için yani sadece veri girdisi (input) için kullanılır.
`ios::out`	Dosya yazmak için yani dosyaya sadece veri göndermek (output) için kullanılır.
`ios::trunc`	Mevcut bir dosya varsa açmadan önce içi boşaltılır (truncate). .

Tablo 24. Dosya Açma İşlem Modları

Örneğin `cikti.txt` adlı bir dosyayı her seferinde içini boşaltarak sadece yazmak için akış nesnesi aşağıdaki gibi açılabilir ve akış nesnesiyle açılan dosyaya aynı `std::cout` nesnesinde olduğu gibi akışa veri ekleme (stream insertion operator) işleci olan `<<` ile veri ekleyebilir veya `std::cin` nesnesinde oluğu gibi akıştan veri çıkarma işleci (stream extraction operator) olan `>>` ile veri okuyabilirsiniz.

```
ofstream outfile;
outfile.open("cikti.txt", ios::out | ios::trunc );
if(outfile.is_open()){
    os << "Hello World!";
}
```

Benzer şekilde, aynı dosyayı hem okuma ve hem de yazma amacıyla aşağıdaki gibi açabilirsiniz;

```
fstream  afile;
afile.open("cikti.txt", ios::out | ios::in );
if (!afile.is_open()) {
    throw CustomException(afile, "Dosya Açılamadı!");
}
```

Çıktı dosyası akışında `<<` işleci yerine, üye fonksiyon olan `write()` da kullanılabilir:

```
if(afile.is_open()){
    char metin[] = "İlhan";
    os.write(metin, 3); // Metinden 3 characters akışa yazılıyor
}
```

Dosyadan veri okuma aşağıdaki şekilde yapılabilir;

```
/*Metin.txt Dosasında aşağıdakilerin olduğunu varsayalım;
İlhan Özkan 25 4 6 1987
Mehmet Yıldırım 15 5 24 1976
```

```
*/
std::ifstream is("metin.txt");
std::string adi, soyadi;
int yas, dogumAyi, dogumGunu, dogumYili;
while (is >> adi >> soyadi >> yas >> dogumAyi >> dogumGunu >> dogumYili)
   cout << adi << " " << soyadi;
```

Bir C++ programından çıkıldığında otomatik olarak tüm akışları temizler, ayrılmış belleği serbest bırakır ve açık tüm dosyaları kapatır. Ancak bir programcının programı sonlandırmadan önce açık olan tüm dosyaları kapatması her zaman iyi bir uygulamadır.

Aşağıda `fstream`, `ifstream` ve `ofstream` nesnelerinin üyesi olan `close()` yönteminin kullanılarak dosyaları kapatabiliriz.

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

int main () {
    string adiSoyadi;
    int yas;
    char cinsiyet;

    cout << "Adınızı Giriniz: ";
    getline(cin, adiSoyadi);
    cout << "Yaşınızı Giriniz: ";
    cin >> yas;
    cout << "Cinsiyetinizi Giriniz (K-E): ";
    cin >> cinsiyet;

    //Program bilgilerini dosyaya yazalım:
    ofstream outfile;  // akış nesnesi tanımlandı.
    outfile.open("output.txt"); //dosya açıldı
    outfile << "Bu dosya Yaş ve Cinsiyet Bilgilerini İçerir" << endl;
    outfile << "Adı:" << adiSoyadi << endl;
    outfile << "Yaşı:" << yas << endl;
    outfile << "Cinsiyeti:" << cinsiyet << endl;
    outfile.close(); // dosya kapandı

    //Yardığımız dosyayı okuyalım:
    string satir;
    ifstream infile; // akış nesnesi tanımlandı.
    infile.open("output.txt"); //dosya açıldı
    getline(infile, satir);
    cout << "Dosyadan Okunan Satır:" << endl << satir << endl;
    getline(infile, satir);
    cout << "Dosyadan Okunan Bir Sonraki Satır:" << endl << satir << endl;
    infile.close(); // dosya kapandı
}
/*Program Çalıştırıldığında:
Adınızı Giriniz: İlhan Özkan
Yaşınızı Giriniz: 50
Cinsiyetinizi Giriniz (K-E): E
Dosyadan Okunan Satır:
Bu dosya Yaş ve Cinsiyet Bilgilerini İçerir
Dosyadan Okunan Bir Sonraki Satır:
Adı:İlhan Özkan

...Program finished with exit code 0
```

```
*/
```

Program çalıştığında oluşan `output.txt` metin dosyasının içeriği aşağıda verilmiştir;

```
Bu dosya Yaş ve Cinsiyet Bilgilerini İçerir
Adı:İlhan Özkan
Yaşı:50
Cinsiyeti:E
```

Buraya kadar işlediğimiz akışlar konsola yazdığımız veya klavyeden okuduğumuz gibi metin olarak çalışır. Yani oluşan dosyalar gözle okunabilir metinlerdir.

Çalışılan dosyalar her zaman metin dosyası olamayabilir. Bir nesneyi bellekte olduğu gibi yani ikili olarak aynen dosyaya saklamak isteyebiliriz. Bu durumda akışları ikili dosya yazıp okuyacak şekilde açıp işlem yapmalıyız. Bu durumda dosya açma modu olarak `ios::binary` kullanılmalıdır.

Aşağıda ikili olarak oluşturulan bir dosya örneği verilmiştir.

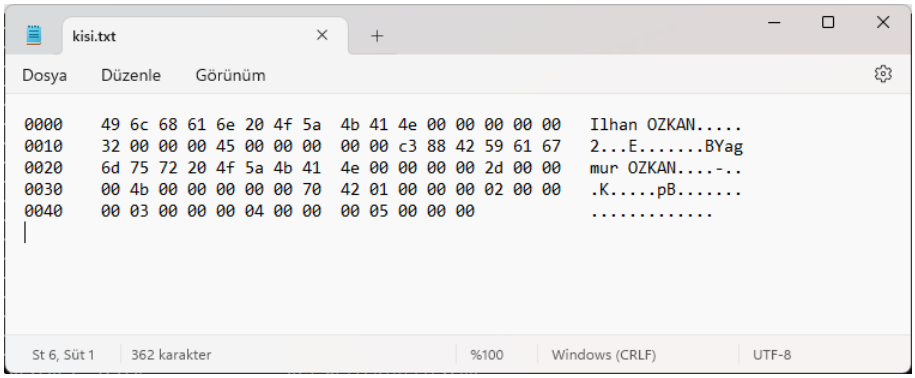
```
#include <iostream>
#include <fstream>
using namespace std;
struct kisi {
    char adiSoyadi[16];
    int yas;
    char cinsiyet;
    float kilo;
};
typedef struct kisi Kisi;

int main() {
    Kisi kisi1={"Ilhan OZKAN",50,'E',100.0};
    Kisi kisi2={"Yagmur OZKAN",45,'K',60.0};
    int dizi[5]={1,2,3,4,5};

    ofstream os("kisi.bin", std::ios::binary);
    if (!os) {
        cerr << "Dosya Açılamadı!" << std::endl;
        return 1;
    }
    os.write(reinterpret_cast<char*>(&kisi1), sizeof(Kisi));
    os.write(reinterpret_cast<char*>(&kisi2), sizeof(Kisi));
    os.write(reinterpret_cast<char*>(&dizi), sizeof(dizi));
    os.close();
}
```

Oluşan `kisi.bin` dosyasına ilk önce iki adet kişi yapısı kaydedilmiştir. Sonrasında 5 tamsayıdan oluşan dizi kaydedilmiştir. Bellekte saklandığı şekliyle dosyaya kaydedildiğinden içeriği ikili olduğundan ikili dosyaları açan görüntüleyiciler ile açılıp görülebilir.

Bir ikili dosyayı onaltılık rakamlara çevirip metin olarak görüntülemek için Windows ortamında aşağıdaki komut verilebilir;

```
C:\Users\ILHANOZKAN>certutil -encodehex kisi.bin kisi.txt
Input Length = 77
Output Length = 367
CertUtil: -encodehex command completed successfully.
```

Artık dönüştürülen dosyayı (`kisi.txt`) istediğimiz metin düzenleyicisinde açıp okuyabiliriz.



Şekil 24. Programın Oluşturduğu İkili Dosya İçeriği

Aynı dosyayı ikili olarak okuyan program aşağıdaki şekilde yazılabilir;

```
#include <iostream>
#include <fstream>
using namespace std;
struct kisi {
    char adiSoyadi[16];
    int yas;
    char cinsiyet;
    float kilo;
};
typedef struct kisi Kisi;

int main() {
    Kisi kisi1, kisi2;
    int dizi[5];

   ifstream is("kisi.bin", std::ios::binary);
   if (!is) {
      cerr << "Dosya Açılamadı!" << std::endl;
      return 1;
   }
   is.read(reinterpret_cast<char*>(&kisi1), sizeof(Kisi));
   cout << "Okunan Kisi Yapısı:" << endl
        << "Kişi Adı:" << kisi1.adiSoyadi << endl
        << "Kişi Yaşı:" << kisi1.yas << endl
        << "Kişi Cinsiyeti:" << kisi1.cinsiyet << endl
        << "Kişi Kilo:" << kisi1.kilo << endl;
   is.read(reinterpret_cast<char*>(&kisi2), sizeof(Kisi));
   cout << "Okunan Kisi Yapısı:" << endl
        << "Kişi Adı:" << kisi2.adiSoyadi << endl
        << "Kişi Yaşı:" << kisi2.yas << endl
        << "Kişi Cinsiyeti:" << kisi2.cinsiyet << endl
        << "Kişi Kilo:" << kisi2.kilo << endl;
   is.read(reinterpret_cast<char*>(&dizi), sizeof(dizi));
   cout << "Okunan Dizi:" << dizi[0] << "," << dizi[1] << ","
        << dizi[2] << "," << dizi[3] << "," << dizi[4] << endl;
   is.close();
}
/* Program Çalıştığında:
Okunan Kisi Yapısı:
Kişi Adı:Ilhan OZKAN
Kişi Yaşı:50
Kişi Cinsiyeti:E
Kişi Kilo:100
Okunan Kisi Yapısı:
Kişi Adı:Yagmur OZKAN
```

```
Kişi Yaşı:45
Kişi Cinsiyeti:K
Kişi Kilo:60
Okunan Dizi:1,2,3,4,5

...Program finished with exit code 0
*/
```

Eğer sadece dizili okumak isteseydik dosya konum göstericisini okuyacağımız yere konumlandırmak gerekir;

```
#include <iostream>
#include <fstream>
using namespace std;
struct kisi {
    char adiSoyadi[16];
    int yas;
    char cinsiyet;
    float kilo;
};
typedef struct kisi Kisi;

int main() {
    Kisi kisi1, kisi2;
    int dizi[5];

   ifstream is("kisi.bin", std::ios::binary);
   if (!is) {
      cerr << "Dosya Açılamadı!" << std::endl;
      return 1;
   }
   is.seekg(2*sizeof(Kisi),ios::beg);
   // dizi iki kişi kaydı sonrasında kaydedilmişti.
   is.read(reinterpret_cast<char*>(&dizi), sizeof(dizi));
   cout << "Okunan Dizi:" << dizi[0] << "," << dizi[1] << ","
        << dizi[2] << "," << dizi[3] << "," << dizi[4] << endl;
   is.close();
}
```

Hem `istream` hem de `ostream` başlığı ikili olarak açılan dosya için dosya konum göstericisini yeniden konumlandırmak için üye fonksiyonları sağlar. Bunlar `istream` için konumu öğrenmede (seek get) kullanılan `seekg()`ve `ostream` için yeniden konumlandırma (seek put) için `seekp()` fonksiyonlarıdır.

Bu fonksiyonlarda konumlandırmanın yönü, bir akışın başlangıcına göre konumlandırma için `ios::beg` (varsayılan), bir akıştaki geçerli konuma göre konumlandırma için `ios::cur` veya bir akışın sonuna göre konumlandırma için `ios::end` kullanılır.

```
streamObject.seekg( n ); // dosya başından itibaren n inci bayt ilerle (varsayılan ios::beg)
streamObject.seekg( n, ios::cur ); // mevcut konumdan itibaren n bayt ilerle
streamObject.seekg( n, ios::end ); // dosya sonundan itibaren n bayt geri gel
streamObject.seekg( 0, ios::end ); // pdosya sonuna ilerle.
```

std::cerr Nesnesini Dosyaya Yönlendirme

Hataları izlemek için kullanılan `std::cerr` nesnesi genellikle konsola hata çıkışı vermek için kullanılsa da bir dosyaya da yönlendirilebilir. Bu, bir programda hataları günlüğe kaydetmeniz gerektiğinde yararlı olabilir.

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

```
int main() {

    ofstream errorLog("log.txt");
    cerr.rdbuf(errorLog.rdbuf());
    cerr << "Hata mesajları bu dosyaya yazılacak!." << endl;
    errorLog.close();
}
```

stringstream Sınıfı

C++ dilinde `stringstream` sınıfı birden fazla dizgiyi aynı anda girdi olarak almak için kullanılır. Kısaca dosya yerine bir metni akış olarak kullanır. Aşağıda örnek bir uygulama verilmiştir;

```
#include <iostream>
#include <sstream>
#include <string>
using namespace std;
int main() {
    string metin = " Merhaba Ilhan OZKAN. Nasılsınız? ";
    stringstream metinAkisi(metin);
    string sozcuk;
    while (metinAkisi >> sozcuk) {
        cout << sozcuk << endl;
    }
}
/* Program Çalıştığında:
Merhaba
Ilhan
OZKAN.
Nasılsınız?

...Program finished with exit code 0
*/
```

Dosyanın tamamın bir `stringstream` nesnesine konulabilir;

```
std::ifstream f("metin.txt");
if (f)
{
    std::stringstream buffer;
    buffer << f.rdbuf();
    f.close();
}
```