BASİT GİRİŞ ÇIKIŞ İŞLEMLERİ

Modüler Programlama

C++ programlama dili, yapısal olmasının yanında aynı zamanda modüler bir programlama dilidir. Yapısal programlamanın çerçevesi *Yapısal Programlama* altında anlatılmıştır. Yapısal programlamada yazılan fonksiyonlar gruplar halinde başka kod dosyalarına konulur ve bun bileşenler gerektikçe projemize `#include` ön işlemci yönergeleriyle (preprocessor directive) dahil edilir.

Modüllere ayırmanın soyutlama (abstraction), değişim yönetimi (change management) ve yeniden kullanım (reusing) gibi üstünlükleri vardır. Modüllere ayrılan bir kodun bakımı da daha kolaydır. Ancak daha fazla fonksiyonun çağrılması, işlemciyi yorar. Çünkü her fonksiyon çağrısında işlemcinin mevcut durumu ve kaydediciler belleğe itilir ve fonksiyondan geri dönüldüğünde eski duruma dönmek içi bu veriler tekrar geri çekilir. İşlemciye yüklenen bu çağırma yükü (calling overhead) günümüz bilgisayarlarında oldukça ihmal edilebilir seviyededir. Ancak milisaniyeler bazında yapılması gereken bir iş varsa böyle zaman kritik işler gerçekleştirmek için bu durum göz önüne alınmalıdır.

C++ programlama dilinde en çok kullanacağımız modüllerden biri de giriş çıkış işlemlerinin (input output-IO) tanımlı olduğu `iostream` başlık (header) dosyasıdır. C dilinde `stdio.h` olarak verilen başlık dosyası yerine nesnelerle tasarlanmış `iostream` başlık dosyası kullanılır. Bu modülü kodumuza dahil etmek (include) için kodu yazdığımız metin editörünüzde kaynak kodun başına `#include` ön işlemci yönergesini (preprocessor directive) aşağıdaki şekilde ekleriz.

```
#include <iostream>
```

Böylece konsola (konsol kelimesi UNIX/Linux işletim sitemindeki terminal ya da Windows işletim sistemindeki komut satırını ifade eder) bir şey yazmak için cin veya klavyeden bir şey okumak ve bir değişkene atamak için ise `cout` nesnelerini kullanabilir hale geliriz.

C programlama dilinde her işletim siteminde çalışan standart birçok başlık dosyası bulunmaktadır. Ancak bunların dışında çalıştığı işletim sistemine özel olan başlık dosyaları da bulunmaktadır;

Başlık Dosyası	Açıklama
`<iostream>`	`std::cout`, `std::cin` nesneleri kullanılarak, konsol ve klavye gibi akış (stream) nesneleri ile giriş ve çıkış işlemleri için nesne ve `std::endl` gibi
`<cmath>`	`sqrt()` , `log2()` , `pow()` gibi matematiksel işlemleri yapmak için kullanılan sınıfı içerir.
`<cstdlib>`	`malloc()` ve `free()` gibi bellek tahsisi ve sistemle ilgili `exit()` ve `rand()` gibi işlevleri içerir.
`<vector>`	Dinamik diziler (vektörler) için konteyner (container) sınıf tanımlarını içerir.
`<string>`	Dizgi işleme için sınıf ve işlevler sağlar `std::string`
`<iomanip>`	Giriş çıkış işlemlerinde değişkenlerdeki ondalık basamak sayısını sınırlamak için `set()` ve `setprecision()` fonksiyonlarına erişmek için kullanılır.
`<cerrno>`	`errno()`, `strerror()`, `perror()` gibi istisna işleme işlemlerini gerçekleştirmek için kullanılır.
`<time>`	`setdate()` ve `getdate()` gibi `date()` ve `time()` ile ilgili fonksiyonlarla işlem yapmak için kullanılır. Ayrıca sistem tarihini değiştirmek ve CPU zamanını almak için de kullanılır.

Tablo 15. En çok kullanılan başlık dosyaları

Konsolda Veri Okuma ve Yazma

C++ dilinde `std::cin` nesnesi, varsayılan olarak klavye ile ilişkilendirilen standart giriş akışı (input stream) olan `stdin` akışından gelen girdiyi kabul etmek için kullanılan `istream` sınıfında yer alır. Akışları, C dilindeki dosyalar olarak düşünebiliriz.

Girdi olarak kullanılan akışlarda, akıştan veri çıkarma işleci (stream extraction operator) (`>>`) ile verileri akıştan çıkarmak ve verilen değişkene eklemek için `cin` nesnesi ile birlikte kullanılır.

```
#include <iostream>
int main() {
   int yas;
   float agirlik;
   std::cin >> yas;
   std::cin >> agirlik;
}
```

Benzer şekilde C++ dilinde `std::cout` nesnesi, varsayılan olarak konsola ile ilişkilendirilen standart çıkış akışı (output stream) olan `stdout` akışına gönderilen çıktıyı kabul etmek için kullanılan `ostream` sınıfında yer alır. Çıktı olarak kullanılan akışlarda, akışa veri ekleme işleci (stream insertion operator) (`<<`) ile verileri verilen değişkeni kullanarak akışa yazmak için `cout` nesnesi ile birlikte kullanılır.

```
#include <iostream>
int main() {
   int yas;
   float agirlik;
   std::cout << "Yaş Giriniz:";
   std::cin >> yas;
   std::cout << "Ağırlık Giriniz:";
   std::cin >> agirlik;
   std::cout << "Girilen Yaş:"<< yas <<" ve Ağırlık:" <<agirlik;
}
/* Program çalıştırıldığında:
Yaş Giriniz:12
Ağırlık Giriniz:75.6
Girilen Yaş:12 ve Ağırlık:75.6

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

Dahil edilen `iostream` başlığı birçok nesne ve bildirim içerir. Bunlardan standart giriş nesnesi olan `cin` ve standart çıkış nesnesi olan `cout` nesnesi `std` adlı aynı isim uzayında (namespace) yer alırlar. Programda bu isim uzayını hep yazmamak için `using namespace std;` talimatı yazılabilir. Aksi halde `cout` veya `cin` nesnesine ulaşmak için yukarıdaki örnekte olduğu gibi iki tane iki nokta üst üste karakteri olan kapsam çözümleme işleci (scope resolution operator) kullanılır. İsim uzayları sonradan anlatılacaktır. Bu durumda aynı kod aşağıdaki gibi kısaltılabilir;

```
#include <iostream>
using namespace std;
int main() {
   int yas;
   float agirlik;
   cout << "Yaş Giriniz:";
   cin >> yas;
   cout << "Ağırlık Giriniz:";
   cin >> agirlik;
   cout << "Girilen Yaş:"<< yas <<" ve Ağırlık:" <<agirlik;
}
```

Benzer şekilde yarıçapı gerçek sayı olarak klavyeden alınan bir çemberin ve çevresini hesaplayan program aşağıda verilmiştir.

```
#include <iostream>
#define PI 3.1415
using namespace std;
int main() {
   float cemberinCevresi;
   cout << "Çemberin Çevresini Giriniz:";
```

```
   cin >> cemberinCevresi;
   float cemberinYariCapi=cemberinCevresi/2/PI;
   cout << "Çevresini Girdiğiniz Çemberin Yarıçapı:"<<cemberinYariCapi;
}
```

Uluslararası Metinler

C++ dilinde `wchar_t`, desteklenen en geniş karakter setinin (UTF-16) tüm karakterlerini temsil edebilecek kadar büyük bir tamsayı türüdür. Normalde, `char` karakter türünden daha büyük bir boyuta sahip olduğundan, ASCII 255 üzerinde karakterleri depolamanız gerektiğinde kullanılır.

```
#include <iostream>
using namespace std;
int main()
{
    const wchar_t message_chinese[] = L"你好，世界";// Chinese for "hello, world"
    const wchar_t message_hebrew[] = L" שלום עולם "; //Hebrew for "hello, world"
    const wchar_t message_russian[] = L"Привет мир"; //Russian for "hello, world"
    const wchar_t message_turkish[] = L"Merhaba Dünya"; //Türkçe "Merhaba Dünya"
    const wchar_t message_turkish2[] = L"ıİüÜğĞiİşŞçÇöÖ"; //Türkçe "Diğer Karakterler
    std::locale::global(std::locale(""));
    std::wcout.imbue(std::locale());
    setlocale(LC_ALL, "Chinese");
    wcout << message_chinese << endl;
    setlocale(LC_ALL, "Hebrew");
    wcout << message_hebrew << endl;
    setlocale(LC_ALL, "Russian");
    wcout << message_russian << endl;
    setlocale(LC_ALL, "Turkish");
    wcout << message_turkish << endl;
    wcout << message_turkish2 << endl;
}
/*Program Çıktısı:
你好，世界
 שלום עולם
Привет мир
Merhaba Dünya
ıİüÜğĞiİşŞçÇöÖ
*/
```

std::iomanip

Bu başlık `std::cin` ve `std::cout` ile ileride göreceğimiz akış nesnelerinde girdi ve çıktıyı biçimlendirmek için kullanılır; Aşağıdaki örnekte `std::setprecision` ile çıktıya gönderilecek hane sayısı belirleniyor;

```
#include <iostream>
#include <iomanip>
#include <cmath>
#include <limits>
int main()
{
    const long double pi = std::acos(-1.L);
    std::cout << "default precision (6): " << pi << '\n'
    << "std::precision(10): " << std::setprecision(10) << pi << '\n'
    << "max precision: "
    << std::setprecision(std::numeric_limits<long double>::digits10 + 1)
    << pi << '\n';
}
/*Program Çıktısı:
default precision (6): 3.14159
std::precision(10): 3.141592654
```

```
max precision: 3.141592653589793239
*/
```

Aşağıda `std::setw` ile yazdırılacak genişlik ve `std::setfill` ile doldurulacak karakter örneği verilmiştir;

```
#include <iostream>
#include <iomanip>
int main()
{
    int val = 10;
    std::cout << val << std::endl;
    std::cout << std::setw(10) << val << std::endl;
    std::cout << "default fill: " << std::setw(10) << 42 << '\n'
    << "setfill('*'): " << std::setfill('*')
    << std::setw(10) << 42 << '\n';
}
/* Program Çıktısı:
10
        10
default fill:         42
setfill('*'): ********42
*/
```

Giriş çıkış işlemlerinde `out<<setiosflags(mask)` veya `in>>setiosflags(mask)` kullanıldığında akışın tüm biçim bayraklarını maskenin belirttiği şekilde dışarı veya içeri ayarlar. Tüm `std::ios_base::fmtflags` listesi:

- `dec` - tamsayı G/Ç için ondalık taban kullan
- `oct` - tamsayı G/Ç için sekizlik taban kullan
- `hex` - tamsayı G/Ç için onaltılık taban kullan
- `basefield` - `dec`|`oct`|`hex`|`0` maskeleme işlemleri için kullanışlıdır
- `left` - sol ayarlama (sağa dolgu karakterleri ekler)
- `right` - sağ ayarlama (sola dolgu karakterleri ekler)
- `internal` - dahili ayarlama (dahili olarak belirlenmiş noktaya dolgu karakterleri ekler)
- `adjustfield` - `left`|`right`|`internal`. Maskeleme işlemleri için kullanışlıdır
- `scientific` - bilimsel gösterim veya `fixed` ile birleştirildiğinde `hex` gösterimi kullanarak kayan nokta türleri oluşturur
- `fixed` - `fixed` gösterimi veya `scientific` ile birleştirildiğinde `hex` gösterimi kullanarak kayan nokta türleri oluşturur
- `floatfield` - `scientific`|`fixed`|(`scientific`|`fixed`)|`0`. Maskeleme işlemleri için kullanışlıdır
- `boolalpha` – alfa sayısal biçimde `bool` türünü ekler ve çıkarır
- `showbase` - tamsayı çıktısı için sayısal tabanı belirten bir önek oluşturur, para birimi göstergesini gerektirir
- `showpoint` - kayan nokta sayı çıktısı için koşulsuz olarak ondalık nokta karakteri oluşturur
- `showpos` - negatif olmayan sayısal çıktı için + karakteri oluşturur
- `skipws` - belirli girdi işlemlerinden önce öndeki boşlukları atlar
- `unitbuf` her çıktı işleminden sonra çıktıyı temizler
- `uppercase` - belirli çıktı çıktılarında belirli küçük harfleri büyük harf eşdeğerleriyle değiştirir

```
#include <iostream>
#include <string>
#include<iomanip>
int main()
{
    int l_iTemp = 47;
    std::cout<< std::resetiosflags(std::ios_base::basefield);
    std::cout<<std::setiosflags( std::ios_base::oct)<<l_iTemp<<std::endl; //57
    std::cout<< std::resetiosflags(std::ios_base::basefield);
    std::cout<<std::setiosflags( std::ios_base::hex)<<l_iTemp<<std::endl; //2f
    std::cout<<std::setiosflags( std::ios_base::uppercase)<<l_iTemp<<std::endl; //2F
```

```
    std::cout<<std::setfill('0')<<std::setw(12);
    std::cout<<std::resetiosflags(std::ios_base::uppercase);
    std::cout<<std::setiosflags( std::ios_base::right)<<l_iTemp<<std::endl; //00000000002f
    std::cout<<std::resetiosflags(std::ios_base::basefield|std::ios_base::adjustfield);
    std::cout<<std::setfill('.')<<std::setw(10);
    std::cout<<std::setiosflags( std::ios_base::left)<<l_iTemp<<std::endl; //47........
    std::cout<<std::resetiosflags(std::ios_base::adjustfield)<<std::setfill('#');
    std::cout<<std::setiosflags(std::ios_base::internal|std::ios_base::showpos);
    std::cout<<std::setw(10)<<l_iTemp<<std::endl; //+#######47
    double l_dTemp = -1.2;
    double pi = 3.14159265359;
    std::cout<<pi<<" "<<l_dTemp<<std::endl; //+3.14159 -1.2
    std::cout<<std::setiosflags(std::ios_base::showpoint)<<l_dTemp<<std::endl; //-1.20000
    std::cout<<setiosflags(std::ios_base::scientific)<<pi<<std::endl; //+3.141593e+00
    std::cout<<std::resetiosflags(std::ios_base::floatfield);
    std::cout<<setiosflags(std::ios_base::fixed)<<pi<<std::endl; //+3.141593
    bool b = true;
    std::cout<<std::setiosflags(std::ios_base::unitbuf|std::ios_base::boolalpha)<<b; //true
    return 0;
}
```