Kütüphane Desteği

Murat A. Kurt, info@basit.web.tr

Murat A. Kurt
info@basit.web.tr
www.basit.web.tr

ENDEKS

(A) Giriş

Bu PDF dosyasında Kütüphaneler tanıtılıyor ve avantajları inceleniyor. Günüzümde en yaygın yazılım geliştirme tekniği parçalı uygulamalardır. Parçalı olmalarının nedeni ise, uygulamanın birçok birbirinden bağımsız dosyalardan oluşmasıdır. Ana dosya uygulamanın kendisidir ve diğer tüm dosyalar kütüphane dosyalarıdır.

Eğer uygulamada bir güncelleme yapılacak ise veya kritik hatalar düzeltilecek ise, tüm uygulamayı yeniden kodlamak veya her satırının tek tek incelemek gerekmez. Güncelleme yapılacak kütüphane dosyaları veya onarılması gereken kütüphane dosyalarını değiştirmek yeterlidir.

Kütüphane tekniği sayesinde uygulamaları daha iyi geliştirebilir ve dağıtabilirsin. Ayrıca profesyonel ve çok kapsamlı uygulamalar ancak kütüphane tekniği ile gerçek manada mümkün oluyor. Herşeyi tek bir uygulama dosyasına sığdırmaya çalışmak her zaman mümkün olmadığı gibi, her zaman en iyi çözüm değildir.

(B) Kütüphane Nedir?

Bir kütüphane paket gibidir. İçinde birçok yapı taşı, obje, bileşen, kaynak ve veriler içerebilir. Fakat kendi başına bir uygulama değildir. Daha çok bir uygulama tarafından kullanılabilecek bir komut seti gibidir.

Başka bir değimlede gerçekten bir kütüphane gibidir. İçerdiği her elamanı bir kütüphanede bulunan bir kitap gibidir. Birbirinden farklı amaçlar için tasarlanmış birçok elemanı bir arada sunabilir.

İşin güzel yönü ise bir kez hazırlanmış ve derlenmiş bir kütüphane dosyası birçok projede kullanılabilir. Böylece sık kullanılan elemanlar bir tek kütüphanede derlenebilir ve nerede ve ne zaman lazım olurlarsa, kullanılabilirler. Ayrıca her seferinde kütüphaneyi tekrar derlemek gerekmez.

Eğer güncelleme yapılacak ise, sadece kütüphane dosyasını yeni sürümüyle değiştirmek yeterlidir. Tüm uygulamayı yeniden kurma zorunluluğu yoktur.

(B) Kütüphane Çeşitleri ve Bağlantı Modelleri

Kütüphane konsept olarak her işletim sisteminde aynı olsada, her işletim sisteminin kendi kütüphane modeli vardır. Mesela Mac için derlenen bir kütüphane dosyası Windows'ta çalışmayacağı gibi, Windows için derlenen bir kütüphane dosyasıda Mac'te çalışmaz.

İşletim sistemlerine göre ayrı ayrı kütüphane modelleri olsada, Basit, Basit yazılım diliyle geliştirilen uygulamaların kullanabileceği, işletim sisteminden bağımsız özel bir kütüphane modeli daha sunmaktadır. Bu özel kütüphane modeli "Basit Eklentiler Kütüphanesi" sadece Basit ile geliştirilebilir ve sadece Basit ile geliştirilen uygulamalar tarafından kullanılabilir. Bu kütüphaneler Mac, Linux veya Windows için ayrı ayrı değildir, üniversaldir. Yani bir defa derlenmiş bir "Basit Eklentiler Kütüphanesi" her üç işletim sisteminde ister 32Bit, ister 64Bit uygulamalarda aynen kullanılabilir.

Fakat Mac, Linux veya Windows için derlenen kendi kütüphane modelleri 32Bit uygulama için 32Bit olarak derlenmeli ve sunulmalı ve 64Bit uygulama için yine 64Bit olarak derlenmeli ve sunulmalıdır. Aksi taktirde uygulama hata verecektir veya çökecektir veya hiç açılamayacaktır.

Çeşitler:

İşletim Sistemi	Dosya Eki	Açılımı
Üniversal	BEK	Basit Eklentiler Kütüphanesi
Mac	DyLib	Dinamik Kütüphane (Dynamic Library)
	SO	Paylaşılabilir Obje (Shared Object)
Linux	SO	Paylaşılabilir Obje (Shared Object)
Windows	DLL	Dinamik Bağlanılabilir Kütüphane (Dynamic Link Library)

Makineye göre derleme durumu:

BEK	DyLib	Mac So	Linux So	DLL
Anlamsız, gerekmiyor	32Bit için ayrı, 64Bit için ayrı zorunludur	32Bit için ayrı, 64Bit için ayrı zorunludur	32Bit için ayrı, 64Bit için ayrı zorunludur	32Bit için ayrı, 64Bit için ayrı zorunludur

Kütüphane bağlantıları statik veya dinamik olarak gerçekleştirilebilir. Statik bağlantılı kütüphaneler uygulamanın açılışı esnasında önceden belleğe okunur ve aktifleştirilir ve sonra uygulama açılır. Statik bağlantı uygulama sonlandırılana kadar kaldırılamaz, ancak uygulama sonlandırılınca otomatik olarak kaldırılır. Dinamik bağlantılı kütüphaneler açılmış bir uygulamaya sonradan bağlanabilir ve uygulama sonlandırılmadan tekrar kaldırılabilir.

(B) Basit Eklentiler Kütüphanesi

Bu tür kütüphaneler sadece Basit ile geliştirilebilir ve sadece Basit ile geliştirilen uygulamalar tarafından kullanılabilir. Bu tür kütüphanelerin avantajları şöyle:

Statik bağlantıda sunulan elemanlar:	Dinamik bağlantıda sunulan elemanlar:
• Global birimde bulunan Değişkenler, Dizilimler, ProsedürBağları, İşaretler • Yapılar • Objeler ve Bileşenler • Prosedürler • Görsel Kaynaklar • Metinsel Kaynaklar • Salt Kaynaklar • Salt Veri Alanları	• Global birimde bulunan Değişkenler, Dizilimler • Prosedürler • Görsel Kaynaklar • Metinsel Kaynaklar • Salt Kaynaklar • Salt Veri Alanları

Diğer avantajları şöyle:

- İşletim sisteminden bağımsızdır
- 32- ve 64Bit için ayrı ayrı derlenmez, fakat her iki derlemede de aynen kullanılabilir
- Unikod desteklidir ve tüm elemanların isimleri Türkçe karakterler içerebilir
- Gerçek kütüphaneler ile Basit uygulamaları arasında ara-katman görevi görebilir

Ara-Katman görevi:

Bu tür kütüphaneler ile gerçek bir DyLib, So veya DLL'i türkçeleştirilebilir. Gerçek kütüphane bir BEK'te statik veya dinamik olarak bağlanabilir ve kendi elemanlarını gerçek kütüphaneden elde ettiği elemanlar üzerine inşa edebilir. Böylece işletim sistemi API komutları ve birçok başka kütüphaneler Basit ortamında Türkçe olarak sunulabilir.

(B) Natif Kütüphaneler

Bu tür kütüphaneler Mac için DyLib ve So, Linux için So ve Windows için DLL dosyalarıdır. Basit gerçek kütüphanelerden sadece Prosedür elemanlarını kullanabiliyor ve maalesef prosedürlerin parametre adeti en fazla altı ile sınırlıdır. Bu tür dosyalar C veya C++ ile geliştirilebilir ve Basit ortamında kullanılabilir.

Avantajları:

- İşletim sistemi komut setlerine tam erişim sağlar (API, Core Foundation, Cocoa Framework, Gtk*, vesaire)
- Assembler ile işlemlerin optimal yapılabilmesini ve Basit'te kullanılabilmesini sağlar
- Binlerce açık kaynak kütüphanelerinin Basit ile kullanılabilmesini sağlar
- Basit'e sınırsız olanaklar sunar

Dezavantajları:

- İşletim sistemine bağlıdır
- 32- ve 64Bit derlemeleri ayrı ayrı yapılmalı ve sunulmalıdır
- Basit ile geliştirilemez, C/C++ Derleyicisi gerektirir
- Unikod desteği tam olarak yoktur ve daha çok ASCII biçiminde metinler kullanılabilir

Konsept:

Basit ortamında sunulması öngörülen işletim sistemi komut setleri veya başka kütüphaneler ek bir kütüphaneyle Basit için uyarlanabilir ve bir "Basit Eklentiler Kütüphanesi" ile Basit ortamında Türkçe elemanlar aracılığıyla sunulabilir. Böylece Basit ortamında yine herşey Türkçe kalabilir.

(B) STDCALL, CCALL/CDECL

Gerçek kütüphanelerin sunduğu elemanları kullanmak için, kütüphaneyi derlerken kullanılan biçimi bilmek gerekiyor.

Microsoft Derleyicileri genelde STDCALL modelini sunar. Visual C++ ile derlenen DLL dosyalarının elemanlarına erişim STDCALL biçiminde yapılır. Standart C biçimi ise CDECL'dir.

Yani genel olarak Windows'ta sunulan DLL'ler STDCALL biçiminde kullanılabilinir, Mac için sunulan DyLib ve So'lar ve Linux için sunulan So'lar genelde CDECL olarak sunulur. Fakat bu illaki böyle olacak değildir.

Mesela birçok açık kaynak kütüphaneleri genelde Linux ortamında geliştirildiği için ve genelde GNU GCC derleyici setiyle derlendiği için CDECL biçiminde kullanılabilir. Böyle bir kütüphane dosyasını derlenmiş haliyle Windows'ta kullanmak istediğimizde yine CDECL biçiminde kullanmak zorundayız.

Eğer kütüphaneyi kullanacağımız zaman STDCALL veya CCALL/CDECL biçimini özel olarak belirtmezsek, o halde Basit sanal motoru Windows'ta STDCALL, Mac ve Linux'ta CDECL biçimiyle kütüphaneyi kullanmaya çalışacaktır. Yanlış biçimde kurulan bağlantı başarısız olur ve bu durum uygulamanın sağlıklı çalışmasına engel olabilir.

Ayrıntılı bilgi “Statik Bağlantı” ve “Dinamil Bağlantı” bölümlerinde verilecektir.

(C) Statik Bağlantı

Statik bağlanan bir kütüphane dosyası uygulama açılmadan önce hazırlanır ve başarılı bağlanırsa, son olarak uygulamanın kendisi açılır. Bu tür bağlantılar sadece global birimde yapılabilir ve modüler biçimde, ithal edilen elemanlarıyla birlikte, tüm uygulama boyunca geçerliliğini korur. Uygulamanın sonlanmasıyla birlikte statik bağlantı otomatik olarak çözülür. Bu tür bağlantı "Kütüphane" bölümüyle yapılır:

```
Kütüphane ModülerAdı Kaynak/Kaynağı 'dosya'
        İthal eleman ...
KütüphaneSon
```

Modüler adı seçmeli bir tanımlamadır. Bu tanımlama altında bağlantı kurulan kütüphanenin ithal edilen elemanları erişime açılıyor. Mesela tanımlama **Mödülüm** ise, bir ithal elemanı mesela **YapBirŞey** ise, elemana erişebilmek için girilen kod "**Modülüm.YapBirŞey**" olur. Bağlantı kurulan "Basit Eklentiler Kütüphanesi" aşağıda belirtilen öğeleri ithal edebiliyor:

- Global birimde bulunan Değişkenler, Dizilimler, ProsedürBağları, İşaretler
- Yapılar
- Objeler ve Bileşenler
- Prosedürler
- Görsel Kaynaklar
- Metinsel Kaynaklar
- Salt Kaynaklar
- Salt Veri Alanları

Elemanlar kütüphanede nasıl tanımlandıysalar, ithal edilirken aynı şekilde ithal edilmek zorundalar. Sıralama önemsizdir, fakat ayrıntıları aynı olmak zorundadır. Örnek:

```
Kütüphane Elemanlarım Kaynağı 'filanca.bek'
        İthal Değişken Rakam rak1; rak2
        İthal Dizilimsel Metin bilgi; not
        İthal İşaret bellekteBirAlan
        İthal Yapı birYapı
        İthal Obje/Bileşen birObje
        İthal ProsedürBağı proBağ ( Bayt a; b; c; Karakter d ) Bayttır
        İthal Prosedür deBirŞey ( Metin neDiyim ) Metindir
        İthal GörselKaynak Resimlerim
        İthal MetinselKaynak Yazılarım
        İthal SaltKaynak Dosyalarım
        İthal SaltVeriAlanı BirTakımVerilerim
KütüphaneSon
```

Değişkenler ve Dizilimler tanımlamaları ve türleri aynı olmak üzere ithal edilebilirler. Yalnız değer girilemez. Aşağıda belirtilen terim yanlıştır ve hata verecektir:

```
...
İthal Değişken Rakam rak1 = 5            ?        YANLIŞ!!!
...
```

Fakat farklı türden değişkenler veya dizilimler normal biçimde belirlenebilir. Örnekler:

```
...
İthal Değişken Rakam rak1; rak2; Virgül vir1; vir2; Karakter harf; Metin birYazı
İthal Dizilimsel Rakam rak1; rak2; Virgül vir1; vir2; Karakter harf; Metin birYazı
İthal Belirle Rakam rak1; rak2; Virgül vir1; vir2; Karakter harf; Metin birYazı
...
```

İşaret tanımı yine aynı tanımlama adı altında yapılmalıdır. İşaretin bellek alanı rezervasyonu kütüphane içinde yapılabilir, fakat ithal esnasında yapılamaz. Aşağıda belirtilen terim yanlıştır ve hata verecektir:

```
...
İthal İşaret bellekteBirAlan Boyut 500      ?        YANLIŞ!!!
...
```

Bir kütüphanede belirtilen yapısal, objesel veya bileşen-tipi tanımlamalar ithal edilebilir:

```
...
İthal Belirle YapıTürü YapıAdı              ? YapıTürü kütüphanede bulunmak zorundadır
...
```

ProsedürBağları ve Prosedürler mutlaka tanımlama isimleriyle, parametre adetleri, parametre türleri ve parametre sıraları, parametre standart değerleriyle ve sonuç verme türleriyle kütüphanedeki tanımlamalarıyla aynı olmak zorundadır. Ancak parametre isimleri aynı olmak zorunda değildir, fakat önerilir. Bu saydıklarımın dışında yukarıda belirttiğim öğelerden herhangi biri yanlış girilirse bağlantı sağlıklı çalışmaz veya hiç çalışmaz veya uygulamanın çökmesine neden olabilir. GörselKaynak, MetinselKaynak, SaltKaynak ve SaltVeriAlanı tanımları kütüphanede nasıl tanımlandıysalar, aynen öyle tanımlanmalıdırlar. Ayrıca öğeleride yine kütüphanede tanımlanan isimleriyle kullanılabilirler.

Tanımlamaların aynı olmasındaki sebep ise şudur:

Basit derlenen kütüphanelerde Erişilir olarak belirtilmiş elemanlar için kütüphane bünyesinde bir İhraç Tablosu (Export Table) oluşturur. Bu tabloda erişimi sağlanacak eleman ve kütüphane bünyesindeki gerçek adresi tutulur. Söz konusu elemana erişmek için, onu temsil eden tanımlama üzerinden İthal komutuyla İhraç Tablosundan adresi elde edilir.

Eğer bağlantı kurulacak kütüphane dosyası natif bir tür ise (DyLib, So, DLL), o halde bağlantı iki şekilde kurulabilir. Birinci yöntem yukarıdaki yöntem ile aynıdır:

```
Kütüphane ModülerAdı Kaynak/Kaynağı 'dosya'
        İthal eleman ...
KütüphaneSon
```

Natif kütüphanelerin elemanlarına erişim kendi bünyelerinde bulunan İhraç Tabloları (Export Tables) tarafından sağlanır. Fakat natif kütüphaneler için İhraç Tablo Biçimi standart değildir ve aslında derleyiciye göre değişir.

Microsoft derleyicileri İhraç Tablosunu STDCALL biçiminde yönetir. Mac ve Linux ortamında yaygın olan derleyiciler ise İhraç Tablosunu CDECL biçiminde yönetir. Fakat bu durum genelde böyle olsada, net değildir. Windows ortamında sunulan her DLL illaki İhraç Tablosunu STDCALL biçiminde sunmayabilir. Mesela birçok açık kaynak kütüphaneleri genelde Linux ortamında geliştirildiği için, Windows'tada GCC ile derlenir ve İhraç Tablosunu CDECL biçiminde sunar.

Basit, İhraç Tablo Biçimi özellikle belirtilmez ise, Windows'ta STDCALL, Mac ve Linux'ta CDECL biçimiyle kütüphane dosyasının İhraç Tablosuna ve elemanlarına erişim sağlamaya çalışacaktır. Eğer bağlantı kurulması istenen kütüphane mesela Windows ortamında CDECL olarak girilmek zorundaysa, o halde yukardaki bağlantı yöntemi yetersiz kalır. Böylesi bir durumda bağlantı yöntemi şöyledir:

```
Kütüphane ModülerAdı Kaynak/Kaynağı 'dosya'; CDECL
        İthal eleman ...
KütüphaneSon
```

Kaynak dosya girildikten sonra bir noktalı virgül gelir ve ardına İhraç Tablosu Biçimi girilir. Geçerli seçenekler şöyledir:

- STDCALL
- CCALL veya CDECL (her ikiside aynıdır)

Böylece Windows'ta CDECL bağlantısıda kurulabilir ve Mac ve Linux'ta STDCALL bağlantısıda kurulabilir.

Natif kütüphaneler sadece prosedür elemanları sunabilirler. Yani girilebilecek tek doğru ithal yöntemi şöyledir:

```
...
İthal Prosedür Adı ( ... ) ...
...
```

İthal edilmesi istenen prosedürün adı aynı İhraçTablosunda geçtiği gibi girilmek zorundadır. Natif kütüphanelerin İhraç Tabloları ASCII karakter setiyle yönetilir. Burada Unikod geçmez.

Natif kütüphaneler Basit ile geliştirilemez, fakat kullanılabilir. Bu tür kütüphaneler genelde C veya C++ ile geliştirilir. C/C++ veri türleri ile Basit veri türleri farklıdır ve bu yüzden ithal edilecek prosedürün parametre türleri ve sonuç verme türü aşağıda belirtilen karşılaştırmaya göre düzenlenmelidir:

Basit Veri Türü	Sonuç Verme Türü	C/C++ Veri Türü
YarıBayt	YarıBayttır	char
Bayt, Ascii	Bayttır, Asciidir	unsigned char
KüçükRakam	KüçükRakamdır	short
ÇiftBayt, Karakter, SafKüçükRakam	ÇiftBayttır, Karakterdir, SafKüçükRakamdır	unsigned short
Rakam	Rakamdır	long
SafRakam, Renk, Saat	SafRakamdır, Renktir, Saattir	unsigned long
EsnekRakam	EsnekRakamdır	int
SafEsnekRakam, Tarih, TarihSaat	SafEsnekRakamdır, Tarihtir, TarihSaattir	unsigned int
KüçükVirgül	KüçükVirgüldür	float
Virgül	Virgüldür	double

Basit'in Metin ve AsciiMetni türleri zaten dizilimdir ve özellikle dizilim olduklarını vurgulamak hata verecektir. Basit'ten C/C++'a “Dizilimsel Metin” türü veri akışı sağlanamıyor. Dizilim olarak belirlenebilir türler şöyledir:

Basit Veri Türü	Sonuç Verme Türü	C/C++ Veri Türü
Dizilimsel YarıBayt	Dizilimsel YarıBayttır	char*
Dizilimsel Bayt, Ascii	Dizilimsel Bayttır, Asciidir	unsigned char*
AsciiMetni	**AsciiMetnidir**	unsigned char*
Dizilimsel KüçükRakam	Dizilimsel KüçükRakamdır	short*
Dizilimsel ÇiftBayt, Karakter, SafKüçükRakam	Dizilimsel ÇiftBayttır, Karakterdir, SafKüçükRakamdır	unsigned short*
Metin	**Metindir**	unsigned short*
Dizilimsel Rakam	Dizilimsel Rakamdır	long*
Dizilimsel SafRakam, Renk, Saat	Dizilimsel SafRakamdır, Renktir, Saattir	unsigned long*
Dizilimsel EsnekRakam	Dizilimsel EsnekRakamdır	int*
Dizilimsel SafEsnekRakam, Tarih, TarihSaat	Dizilimsel SafEsnekRakamdır, Tarihtir, TarihSaattir	unsigned int*
Dizilimsel KüçükVirgül	Dizilimsel KüçükVirgüldür	float*
Dizilimsel Virgül	Dizilimsel Virgüldür	double*

Basit'in bazı veri türlerinin karşılığı C/C++'da yoktur ve bu sebepten dolayı bazı veri türleri natif bağlantılarda ne parametre türü, nede sonuç verme türü olarak kullanılamaz. Bu veri türleri şöyledir:

ProsedürBağı, Esnek, İz, Yapı, Obje, Bileşen

Bunlar dışında "GenişRakam" ve "SafGenişRakam" Basit 1.0 sürümünde C/C++ ile veri akışında desteklenmiyor. Ayrıca "İşaretli" tanımlamalar C/C++ ile uyumlu değildir ve kullanılamaz.

"İşaret" türü tanımlamalar C/C++ ile uyum sağlar, fakat Basit ile özel olarak aynı biçimde kullanılaması gerekir. Bu türün C/C++'ta birçok karşılığı bulunuyor:

char*, unsigned char*, short*, unsigned short*, long*, unsigned long*,
int*, unsigned int*, float*, double*

C/C++'ta hangi biçimde organize ediliyorsa, aynı biçimde Basit'te kullanılması gerekiyor. Aksi taktirde problemlere sebebiyet verebilir.

Eğer prosedürün C/C++'ta sonuç verme türü yoksa, yani türü "void" ise, Basit'te sonuç verme türü girilmez, aksi taktirde uyumlu tür girilmelidir:

Basit	C/C++
Prosedür hesapla (Virgül a; b)	void hesapla (double a, double b)
Prosedür hesapla (Virgül a; b; Ascii c) Virgüldür	double hesapla (double a, double b, unsigned char c)
Prosedür geriVer (Metin a; b) Metindir	unsigned short* geriVer (unsigned short* a, unsigned short* b)
Prosedür isaretiVer (Bayt hangisi) İşarettir	unsigned int* isaretiVer (unsigned char hangisi)
Prosedür tamamdir ()	void tamamdir ()

Örneklerde de görüldüğü gibi, prosedür tanımlama isimlerinde Türkçe karakterler yoktur. Natif kütüphanede nasıl giriliyor ise, aynı şekilde girilmelidir. Hatta karakterlerin büyük ve küçük yazılışlarına bile dikkat edilmelidir. Eğer natif kütüphanede geçen prosedür adı "geriVer" ise, Basit'te yapılan ithal tanımlama adıda aynı olmalıdır. Eğer "geriver" veya "GeriVer" veya "Geriver" gibi farklı bir biçimde girilse, prosedür tanımı doğrulanamaz ve ithal edilemez.

Unikod desteyi C/C++ yazılım dillerinde standartlaştırılmamıştır ve işletim sistemlerinin sunduğu imkanlara bağlıdır. C/C++ standartlarında Unikod için özel bir destek yoktur. Bu yüzden C/C++ prosedürleriyle çalışırken daha çok Ascii karakterleriyle çalışmak gerekir veya işletim sisteminin sunduğu imkanlardan yararlanmalıdır.

Windows birçok API (Windows NT ver üzeri için geçerlidir) prosedürlerini hem Ascii (*_A), hemde Unikod (*_W) olarak sunuyor. Mac ve Linux ise bu konuda özel birşey sunmuyor, fakat metinleri UTF8 biçiminde yönetiyorlar. Windows NT 4 ve üzeri sürümleri için "Metin", eski sürümler için "AsciiMetni" türü kullanılabilir. Mac ve Linux için ise "AsciiMetni" kullanmak daha iyi olur.

(C) Dinamik Bağlantı

Dinamik bağlanan bir kütüphane dosyası uygulama açıldıktan sonra herhangi bir prosedür bölümü içinden kurulabilir ve işi bitince tekrar kaldırılabilir. Dinamik bağlantıları KütüBağ modülü sağlamaktadır. KütüBağ ile kurulan bağlantılar her paralel akımda farklı kimlik üzerinden sunulur. Bağlantı, kaldırılana kadar geçerli kalır.

Eğer bağlantı kurulacak kütüphane bir "Basit Eklentiler Kütüphanesi" türündeyse, şu tür elemanları sunabilir:

- Global birimde bulunan Değişkenler, Dizilimler
- Prosedürler
- Görsel Kaynaklar
- Metinsel Kaynaklar
- Salt Kaynaklar
- Salt Veri Alanları

ProsedürBağı, İşaret, Yapı, Obje ve Bileşenler dinamik bağlantıda sunulamıyor, statik bağlantı gerektiriyor. Eğer kütüphane natif bir tür ise (DyLib, So, DLL) sadece prosedür sunabiliyor.

İlk olarak kütüphane dosyasına KütüBağ modülüyle bağlantı kurulmalıdır. Bunun için kütüphane bağlantısı geçerli olduğu süre boyunca başka dinamik bağlantılarla karıştırılmaması için, bir kimlik ile eşleştirilmesi gerekiyor. İlk adım bir kimlik numarası belirtmektir. Kimlik numarasını YeniKimlik modülü yardımıyla elde edebiliriz:

```
Değişken SafRakam bağlantım = YeniKimlik.Kütüphane()
```

Sonra KütüBağ ile bağlantı kurulur. Eğer bir "Basit Eklentiler Kütüphanesine" bağlanılacak ise, geçerli yöntem şöyledir:

```
Eğer KütüBağ.BağlantıKur( bağlantım; 'dosya' ) = Evet İse
        Denetim 'Bağlantı başarıyla kurulmuştur'
Yoksa
        Denetim 'Bağlantı kurulamadı'
EğerSon
```

Eğer bağlantı kurulacak kütüphane natif bir kütüphane ise ve İhraç Tablosu varsayılan olarak yeterliyse, yukarıdaki yöntem aynen kullanılabilir. Fakat İhraç Tablo Biçimi özellikle belirtilmek zorundaysa, o halde girilecek kod şöyledir:

```
...
Eğer KütüBağ.BağlantıKur( bağlantım; 'dosya'; KütüphaneKipi.StdCall ) = Evet İse
...
```

KütüphaneKipi'nin seçeneklerinden İhraç Tablosu Biçimi elde edilir. Eğer bağlantı kurulduysa, sırada ithal etmek istediğimiz elemanlar var.

Değişkenler, Dizilimler ve İşaretler için önce işaretli karşıtlar tanımlamak gerekiyor:

```
Değişken İşaretli Metin Met1; met2
Dizilim İşaretli Renk RenkListem
```

Sonra KütüBağ ile eşleştirilmelidirler:

```
Met1 = KütüBağ.DeğişkenAdresi( bağlantım; 'AnneAdı' )
Met2 = KütüBağ.DeğişkenAdresi( bağlantım; 'BabaAdı' )
RenkListem = KütüBağ.DizilimAdresi( bağlantım; 'Renklerim' )
```

Örnektede görüldüğü üzere gerçek adres tanımları ve işaretli karşıtların isimleri farklı olabilir, fakat tanımlamaların asılları yine kütüphanede nasıl geçiyorsa, öyle girilmeliki, bulunabilsinler. Böyle bir eşleştirme kullanılmadan önce tekrar denetlenmelidir:

```
Eğer Met <> Boş İse
        Denetim 'Anne adı okunabildi'
Yoksa
        Denetim 'Anne adı okunamadı'
EğerSon
```

Eğer bir GörselKaynak, MetinselKaynak veya SaltKaynak okunacak ise, o halde İşaret türünden bir dizilime ihtiyaç olacaktır:

```
Belirle Dizilimsel İşaret ResimKimlikListesi
```

Ve KütüBağ yardımıyla kaynak adresleri bu dizilime okunur:

```
ResimKimlikListesi = KütüBağ.GörselKaynakAdresi( bağlantım; 'Resimlerim' )
```

Eğer SaltVeriAlanı okunacak ise, o halde sadece bir işaret gerekir:

```
İşaret VeriAlanıAdresi
```

Ve KütüBağ yardımıyla kaynak adres işarete kopyalanır:

```
VeriAlanıAdresi = KütüBağ.SaltVeriAlanıAdresi( bağlantım; 'verilerim' )
```

Prosedür bağlantısı kurulacak ise, statik bağlantı bölümünde ayrıntılı bir şekilde ifade edildiği gibi, dinamik bağlantı içinde prosedür bağları düzenlemek gerekiyor. Eğer kaynak kütüphane natif ise, statik bölümünde Basit ile C/C++ veri türü karşılaştırmaları aynen geçerlidir. Bağlantı kurulması istenen prosedüre uyumlu bir prosedürü bağı belirlemek ilk adımdır:

```
ProsedürBağı hesaplayıcı ( Virgül rak1; rak2; Karakter Yöntem ) Virgüldür
```

ve ardından bir Değişken tanımı yapmak lazımdır:

```
Belirle hesaplayıcı fonksiyonum
```

Ve KütüBağ yardımıyla kaynak adres ile prosedür bağlantısı eşleştirilir:

```
fonksiyonum = KütüBağ.ProsedürAdresi( bağlantım; 'hesapla' )
```

Bağlantının başarılı olup, olmadığını denetlemek her zaman önerilen yöntemdir:

```
Eğer fonksiyonum > 0 İse
        Denetim '2 + 3 = ' & fonksiyonum( 2; 3; '+' )
Yoksa
        Denetim 'Üzgünüm, bağlantıda sorun var!'
EğerSon
```

Kütüphane bağlantısı artık gerekmiyor ise, bağlantıyı kaldırmak gerekiyor:

```
KütüBağ.Kaldır( bağlantım )
```

Eğer bağlantı kaldırılmaz ise, kaldırılana kadar geçerli olacaktır ve arabellekte gereksiz yer işgal edecektir. Ayrıca profesyonel olmamış olur. Bu yüzden işi biten bağlantıları hemen kaldırmayı unutmamalı.

(D) Basit Eklentiler Kütüphanesi Tasarımı

İlk olarak yeni bir proje oluşturulur ve derleme modeli olarak "Basit Eklentiler Kütüphanesi" seçilir. Ardından kod dosyası eklenir ve kod girilmeye başlanır. Önemli ayrıntıları:

- Bir kütüphanenin Başlat Prosedürü asla olmaz
- Bir kütüphanene hiçbir paralel tanımlama yapamaz
- Kütüphanenin dışarı sunacağı tüm öğelerinin başına Erişilir anahtar kelimesi gelir

Erişilir olarak belirtilmemiş öğeler kütüphane içinde işini görür, fakat dışarıdan erişime kapalıdır. Erişim imkanı sunulan öğeler şöyle:

- Global birimde olmak üzere tanımlanan
  - Değişkenler
  - Dizilimler
  - İşaretler
  - ProsedürBağları
- Prosedürler
- Yapılar
- Objeler ve Bileşenler
- GörselKaynaklar
- MetinselKaynaklar
- SaltKaynaklar
- SaltVeriAlanları

Örnekler:

```
Erişilir Değişken Rakam rak1; rak2
Erişilir Dizilimsel Metin liste1; liste2; liste3

Erişilir ProsedürBağı birBağlantıModeli( Bayt a; b; c ) Esnektir

İşaret birBellekAlanRezevasyonu Boyut 40

Erişilir Prosedür tahminEt( Rakam a; b; c ) Rakamdır
        Sonuç Matematik.Rastgele(a;b) + Matematik.Rastgele(b;c)
ProsedürSon

Erişilir GörselKaynak Resimlerim
        Resim1 = '...'
        Resim2 = '...'
        Resim3 = '...'
GörselKaynakSon
```

```
Erişilir Yapı kimlikBilgileri
        Değişken Metin Adı; SoyAdı; TCKimlikNo; SafRakam Yaşı; CiltNo
        Dizilimsel Metin Adresi
YapıSon

Erişilir Obje objem
        ...
ObjeSon

Erişilir Bileşen objem
        ...
BileşenSon
```

Erişilir olarak belirtilmiş öğeler statik veya dinamik bağlantıda tanımlanan isimleri üzerinden bağlanabilirler:

```
Kütüphane ModülerAdı Kaynağı 'dosya'
        İthal Değişken Rakam rak1; rak2
        İthal Dizilimsel Metin liste1; liste2; liste3
        İthal İşaret birBellekAlanRezevasyonu
        İthal ProsedürBağı birBağlantıModeli( Bayt a; b; c ) Esnektir
        İthal Prosedür tahminEt( Rakam a; b; c ) Rakamdır
        İthal GörselKaynak Resimlerim
        İthal Yapı kimlikBilgileri
        İthal Obje objem
KütüphaneSon
```

Kullanımı:

```
x = ModülAdı.rak1 + ModülAdı.rak2
```