İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

VISUAL C++ ve NATURAL'DA HOST ve PC ARASINDA VERİ TRANSFERİ

YÜKSEK LİSANS TEZİ

Mine ÖZEL
509960160011

Tezin Enstitüye Verildiği Tarih : 17 Ocak 2000
Tezin Savunulduğu Tarih : 4 Şubat 2000

Tez Danışmanı : Y.Doç.Dr. Ali ERCENGİZ

Diğer Jüri Üyeleri Prof.Dr. Nüzhet DALFES

Doç.Dr. Metin Orhan KAYA

ŞUBAT 2000

ÖNSÖZ

Bu araştırmada, mainframe ve PC (Personal Computer) arasında veri transferi konusu üzerinde durulmuştur. Mainframe'deki bilginin başka bir sisteme taşınabilirliği mümkün olmadığından, bilgiyi başka bir sisteme ulaştırabilmek için mainframe ile PC arasında haberleşme zorunlu bir eylem haline gelmiştir. Mainframe ve PC arasındaki haberleşmeyi sağlamak için, her iki tarafta da uygulamalar geliştirilmiştir, PC'de oluşturulan bir terminal emülasyon ekranı sayesinde bu iki uygulamanın karşılıklı konuşması sağlanmıştır.

Bu kaynakta, bir ilişkisel veri tabanı olan ADABAS ve onun üzerine kurulu, dördüncü kuşak programlama dili olan NATURAL ile ilgili önemli bilgiler bulunabilecektir. Ayrıca, host ve PC arasında iletişimi sağlayan HLLAPI (High Level Language Application Programming Interface) fonksiyonları hakkında detaylı bilgi verilmektedir. Bu araştırma, PC tarafında kullanılan C++ programlama dilinin class yapısı üzerine incelemeyi de içermektedir.

Bu projeyi oluştururken, bana destek olan THY A.O.'daki çalışma arkadaşlarıma ve her zaman yanımda olan aileme teker teker teşekkür ederim. Ayrıca, bana bilgileriyle katkıda bulunan değerli hocam Y. Doç. Dr. Ali ERCENGİZ'e teşekkürlerimi sunarım. Bu kaynağın gelecekte araştırma yapacak olanlara yararlı olmasını dilerim.

Ocak, 2000 Mine ÖZEL

İÇİNDEKİLER

TABLO LİSTESİ

ŞEKİL LİSTESİ

VİSUAL C++ VE NATURAL'DA
HOST VE PC ARASINDA VERİ TRANSFERİ

ÖZET

Bu tezde, mainframe ve PC arasında haberleşmeyi sağlayan bir uygulama geliştirilmiştir. Uygulama, mainframe'de oluşturulan mesajların sistem bağlantısı olmayan noktalara gönderilmesi ve bu noktalardan çekilen mesajların mainframe'e aktarılması gereğinden yola çıkılarak hazırlanmıştır. Oluşturulan uygulama, mainframe'deki mesajları PC ortamına kaydetmektedir. PC ortamındaki mesajları da mainframe'e göndermektedir. Tezin amacı ve uygulamanın anahatları Bölüm 1'de ayrıntılı olarak sunulmuştur.

Mainframe ve PC arasında iletişimi sağlamak amacıyla, iki tarafta çalışan programlar geliştirilmiştir. Programlar, PC tarafında kurulan terminal emülasyon ekranı sayesinde ortak bir zeminde buluşturulmuştur. Bu ortak zemin 'Protokol Ekranı' olarak adlandırılabilir.

Mainframe tarafında, ADABAS İlişkisel Veritabanı Yönetim Sistemi kullanılmaktadır. ADABAS, kayıt sayısının çok fazla olduğu büyük veritabanı sistemlerinde başarıyla kullanılmaktadır. ADABAS'ın özellikleri Bölüm 2'de açıklanmaktadır.

Veritabanı üzerinde yazılım geliştirme aracı olarak NATURAL programlama dili kullanılmıştır. NATURAL programlama dili, ADABAS veritabanını kullanmak üzere geliştirildiğinden ADABAS'ın tüm özelliklerini etkin şekilde kullanma yeteneğine sahiptir. NATURAL hakkında genel bilgi ve mainframe tarafında yazılan programda kullanılan ifadelerin tanıtımı Bölüm 2'de verilmiştir.

PC tarafında Borland Visual C++ Builder kullanılmıştır. Class yapılarının kullanımıyla gerçekleştirilen dosya işlemleri konusunda C++ programlama dilinin üstünlüğü tartışılmazdır. Visual C++ programlama dili kullanılarak, kullanıcı için Windows ortamında daha görsel bir uygulama oluşturulması amaçlanmıştır. Visual C++ programlama dilinin class yapısı konusuna Bölüm 2'de değinilecektir.

Mainframe ve PC arasındaki iletişimi sağlamak için, PC'deki C++ programından çağrılabilen HLLAPI (High Level Language Application Programming Interface) fonksiyonları kullanılmıştır. Bu fonksiyonlar hakkında geniş bilgi Bölüm 2'de bulunabilecektir.

PC, HLLAPI fonksiyonları aracılığıyla, host'a bağlanan bir terminal operatörü gibi davranabilmektedir. Uygulama başlatılınca, PC'deki C++ programı mainframe tarafındaki NATURAL programını tetikler. Bu işlem, kullanıcının 'BASLA' butonuna basmasıyla başlayarak, makro dosyasının okunup yorumlanmasıyla gerçekleştirilir.

Host tarafındaki program, veritabanında PC'ye gönderilecek mesaj arar. Gönderilecek mesaj varsa, protokol ekranını kullanarak, ilk satırda 'HOSTSEND' (veya 'HOSTEND') anahtar kelimesiyle birlikte mesajla ilgili tüm bilgileri PC'ye gönderir. PC aldığı mesajı kaydeder. Mesajın alındığını belirten 'PCGET' anahtar kelimesini protokol ekranına yazar. Host bu cevabı aldıktan sonra, veritabanındaki mesaj kaydını günceller.

Host bu işlemden sonra, protokol ekranına 'HOSTRCV' yazarak PC'nin göndereceği mesajı bekler. PC bu aşamada, ekran işlemlerini ve mesaj kuyruğu dosyasının kontrolünü yapar. 10 saniye bekledikten sonra, aynı döngüye yeniden başlar. Host'tan 'HOSTRCV' cevabını aldığı sırada host'a gönderilecek mesaj varsa, mesajla ilgili bilgileri protokol ekranından host'a gönderir. PC 'PCEND' anahtar kelimesini protokol ekranına yazana kadar, host PC'nin gönderdiği mesaj bilgilerini alır. Sistem içinde gitmesi gereken adreslere gönderir.

Uygulama bu işlemleri yaparak, bir sonsuz döngü şeklinde devam eder. Kullanıcı 'CIKIS' butonuna basarak çıkmayı isterse, uygulama sona erer. Uygulama detayları Bölüm 3'de anlatılmıştır.

DATA TRANSFER BETWEEN HOST AND PC USING VISUAL C++ AND NATURAL

SUMMARY

Within this thesis, an application which provides a communication between mainframe and PC has been developed. The application has been prepared by starting with the need of sending the messages that were built in the mainframe to the points without system connection and transfering the messages that were sent from these points to the mainframe. The developed application registers the messages sent from mainframe to the PC environment. It sends the messages in PC environment to the mainframe. The aim of this thesis and baselines of the application are presented with details, in Part 1.

It has been intended to provide the communication between mainframe and PC, therefore there has been developed programs at both sides. Programs are met on a shared range using terminal emulation which was installed on PC. This shared range is called 'Protocol Screen'.

On mainframe side, ADABAS Relational Database Management System has been used. ADABAS has been used successfully in systems that have great number of records. ADABAS features have been explained in Part 2.

NATURAL programming language is used as a software developing tool on database. NATURAL programming language has the ability of using ADABAS features efficiently, because it had been developed to use ADABAS database. General information about NATURAL and definitions of the statements that were used in the mainframe program are given in Part 2.

On PC side, Borland Visual C++ Builder has been used. For the file processing done by usage of class structure, the superiority of C++ programming language is doubtless. Visual C++ programming language is used, because developing a familiar application for the user in Windows environment is aimed. There will be touched on the subject class structure of Visual C++ programming language in Part 2.

Aiming to provide the communication between mainframe and PC, program-callable HLLAPI (High Level Language Application Programming Interface) functions are used. General information about these functions can be found in Part 2.

PC can behave like a terminal operator connected to the host, using HLLAPI functions. When the application begins, C++ program on PC side forces the NATURAL program execution on mainframe side. This process is started with user's pushing on 'BASLA' button and continues with reading the macro file and parsing and interpreting the file.

The program on the host side, searches messages to be sent to PC in database. If message is found, sends all the information about the message with 'HOSTSEND' (or 'HOSTEND') keyword at the first line, using protocol screen. PC registers the message taken. It writes the keyword 'PCGET' that indicates it has taken the message. After host received this respond, it updates the message record in database.

After this process, host waits PC to send a message writing 'HOSTRCV' keyword in the protocol screen. At this step, PC makes screen processes and controls the message queue file. After waiting 10 seconds, the same loop starts. After PC receives 'HOSTRCV' respond, if there is a message to be sent to the host, it sends the message text to the host using protocol screen. Host gets the message information sent by PC until PC writes 'PCEND' keyword to the protocol screen. It sends the message to the addresses in the system.

The application continues in an infinite loop making these processes. If user intends to exit by pushing 'CIKIS' button, application ends. Application details are explained in Part 3.

1. GİRİŞ

1.1 Tezin Amacı

Bu tezde, mainframe ve PC arasında haberleşmeyi sağlayan bir uygulama gerçekleştirilmiştir. Türk Hava Yolları MSW (Message Switching System) Projesi dahilinde, mainframe'de oluşturulan bazı mesajların sistem bağlantısı olmayan noktalara gönderilmesi gerekmektedir. Bu noktalarda, mesaj alışverişini sağlayan manual teleks makineleri bulunmaktadır. Aynı şekilde, bu noktalardaki manual teleks makinelerinden ve DHMİ'den (Devlet Hava Meydanları İşletmeleri) çekilen mesajların mainframe'e aktarılması gerekmektedir.

THY Haberleşme Merkezi'nde, teleks mesajlarının PC ortamından manual teleks makinalarına PTT hatları aracılığıyla gönderilmesi ve aynı yolla manual teleks makinalarından gelen mesajların PC ortamına aktarılması için T-BAS 2000 adında bir uygulama kullanılmaktadır. Bu uygulamanın işlevini yerine getirebilmesi için, gönderilecek teleks mesajlarının bir PC'de kayıtlı olmasına ihtiyaç vardır. Bu tezde oluşturulan uygulama, mainframe'deki mesajları PC ortamına kaydetmektedir. T-BAS 2000 uygulamasının PC ortamına aktardığı mesajları da mainframe'e göndermektedir.

Mainframe ve PC arasında iletişimi sağlamak amacıyla, iki tarafta çalışan programlar geliştirilmiştir. Programlar, PC tarafında kurulan terminal emülasyon ekranı sayesinde ortak bir zeminde buluşturulmuştur. Bu ortak zemin 'Protokol Ekranı' olarak adlandırılabilir.

1.2 Kullanılan Yazılım Geliştirme Araçlarının Seçimi

Uygulama mainframe tarafında IBM OS390 işletim sistemi üzerinde, PC tarafında network üzerinde kurulu Windows 95 işletim sistemi üzerinde çalışmaktadır. PC ve mainframe arasındaki bağlantıyı sağlamak amacıyla, PC'ye 3270 kartı takılmış ve Extra! for Windows 95/NT yazılımı yüklenmiştir. Bu yazılım sayesinde PC, hosta bağlı bir terminal gibi görev yapmaktadır. Uygulamanın başarıyla çalışması için, hosta bağlı bir session'ın açık olması gerekmektedir.

Mainframe tarafında, ADABAS Vers. 6.2 İlişkisel Veritabanı Yönetim Sistemi kullanılmaktadır. ADABAS, kayıt sayısının çok fazla olduğu büyük veritabanı sistemlerinde başarıyla kullanılmaktadır. ADABAS'ın özellikleri Bölüm 2'de açıklanmaktadır.

Veritabanı üzerinde yazılım geliştirme aracı olarak NATURAL Vers. 2.3 programlama dili kullanılmıştır. NATURAL programlama dili ADABAS veritabanını kullanmak üzere geliştirildiğinden ADABAS'ın tüm özelliklerini etkin şekilde kullanma yeteneğine sahiptir. NATURAL hakkında genel bilgi ve mainframe tarafında yazılan programda kullanılan ifadelerin (statements) tanıtımı Bölüm 2'de verilmiştir.

PC tarafında Borland Visual C++ Builder kullanılmıştır. Class yapılarının kullanımıyla gerçekleştirilen dosya işlemleri konusunda C++ programlama dilinin üstünlüğü tartışılmazdır. Visual C++ programlama dili kullanılarak, kullanıcı için Windows ortamında daha görsel bir uygulama oluşturulması amaçlanmıştır. Visual C++ programlama dilinin class yapısı konusuna Bölüm 2'de değinilecektir.

Mainframe ve PC arasındaki iletişimi sağlamak için, PC'deki C++ programından çağrılabilen HLLAPI (High Level Language Application Programming Interface) fonksiyonları kullanılmıştır. Bu fonksiyonlar hakkında geniş bilgi Bölüm 2'de bulunabilecektir.

1.3 Uygulamanın Genel Akış Özeti

PC tarafındaki C++ programı tarafından çağrılan HLLAPI fonksiyonları sayesinde PC, host'a bağlanan bir terminal operatörü gibi davranabilmektedir. Bu şekilde davranması sebebiyle, PC'deki C++ programı mainframe tarafındaki programı tetikleme yeteneğine sahiptir. Bu nedenle uygulamanın başlatılabilmesi için önce PC tarafındaki uygulamanın çalıştırılması gerekmektedir. Bunun için kullanıcının 'BASLA' butonuna basması yeterlidir. Bu butonun işlevi, PC tarafındaki programın 'PLAY.MAC' adlı dosyayı okuyup host'la bağlantı kurmasını sağlamak ve host tarafındaki NATURAL programını çalıştırmaktır.

PC tarafından çalıştırılan mainframe'deki NATURAL programı, veritabanında PC'ye gönderilecek mesaj olup olmadığını araştırır. Gönderilecek mesaj varsa, protokol ekranını kullanarak mesajla ilgili tüm bilgileri PC'ye gönderir. Host, PC'ye mesaj gönderdiğini belirtmek için, protokol ekranının ilk satır ve ilk sütunundan başlayarak, 'HOSTSEND' (veya 'HOSTEND') anahtar kelimesini yazar. PC bu sırada host'tan tepki beklemektedir. Host'un cevabıyla PC aldığı mesajı manual teleks makinalarına gönderilmek üzere kaydeder. Mesajın alındığını belirten 'PCGET' anahtar kelimesini protokol ekranına yazar. Host bu cevabı aldıktan sonra, veritabanındaki mesaj kaydını günceller.

Host bu işlemden sonra, protokol ekranına 'HOSTRCV' yazarak PC'nin göndereceği mesajı bekler. PC ise bu sırada, ekran işlemlerini ve mesaj kuyruğu dosyasının kontrolünü yapar. 10 saniye bekledikten sonra, aynı döngüye yeniden başlar. Host'tan 'HOSTRCV' cevabını aldığı sırada host'a gönderilecek mesaj varsa, mesajla ilgili bilgileri protokol ekranından host'a gönderir. PC 'PCEND' anahtar kelimesini protokol ekranına yazana kadar, host PC'nin gönderdiği mesaj bilgilerini alır. Sistem içinde gitmesi gereken adreslere gönderir.

Uygulama bu işlemleri yaparak, bir sonsuz döngü şeklinde devam eder. Kullanıcı 'CIKIS' butonuna basarak çıkmayı isterse, uygulama sona erer.

2. YAZILIM GELİŞTİRME ARAÇLARININ TANITIMI

2.1 ADABAS İlişkisel Veritabanı Yönetim Sistemi

Bu kısımda ADABAS'ın dosya yapısı, kayıt deseni ve descriptor'ların etkin kullanımı özellikleri tanıtılacaktır.

Kütük ve Kayıt Deseni

ADABAS İlişkisel Veritabanı Yönetim Sistemi'nde performansı sağlamak için aşağıdaki teknikler kullanılır:

- Multiple-value (çokdeğerli) alanlar ve periyodik grupların kullanımı
- Bir ADABAS dosyasının birden fazla kayıt tipini içermesi
- Fiziksel kütüklerin tek bir mantıksal (genişletilmiş) kütüğe bağlanması
- Veri çiftlerinin (data duplication) kontrolü

Multiple-Value Alanlar ve Periyodik Gruplar

Periyodik grubun pratik kullanımını göstermesi açısından aşağıdaki örnek yararlı olacaktır:

Tablo 2.1 Periyodik Grup Örneği

Order Name	Order Date	Date Filled	Customer Number	Date Required	Item Code	Quantity
A1234E	29MAR	-	UK432M	10JUN	24801K	200
		-		15APR	30419T	100
		-		01JUN	27395R	300

ORDERS Kütüğü Kaydı:

					Order_item	
			1006		24801K	200
			0106		27395R	300
A1234E	2903	UK432M	1504		30419T	100
Order_no	Ord_date	Cust_no	Req_date	Fill_date	Item_code	Quantity

Tablo 2-1'de gösterilen örnekte, tablodaki sipariş bilgisi ORDERS adındaki ADABAS dosyasındaki kayıt formatına dönüştürülmüştür. Her bir sipariş kaydı çeşitli sayıda sipariş adedine izin veren periyodik grup içermektedir. Bu durumda, ORDER_ITEM periyodik grubu, ITEM_CODE alanını ve bağlantılı kayıtları (QUANTITY, REQ_DATE ve FILL_DATE) kapsayarak, 191'e kadar farklı adet, istene miktar, gereken tarih ve sipariş formunun doldurulduğu tarihi belirtebilir. Her bir adet/miktar/gereken tarih/doldurulan tarih grubu bir 'occurrence' (oluş) olarak adlandırılır; her bir periyodik grup için 191'e kadar occurrence mümkündür.

Periyodik grubun tek karakteristik özelliği – occurence dizisinin bakımı yeteneği – periyodik grup yapısının seçilme nedenidir. Bir periyodik grup üç occurrence içeriyorsa ve birinci veya ikinci occurrence sonradan silinirse, bu olaylar boşlukla (null) doldurulur; üçüncü occurrence üçüncü pozisyonda kalır. Bu öndeki boş girişlerin multiple-value alanlarda kullanıldığı yolla çelişir.

ORDERS kütüğü için gösterilen kayıt formatı en mantıklı yol gibi görünmeyebilir; bununla birlikte, boşluk içerebilecek alanlar kayıt sonunda veritabanı alanı tasarrufu için birlikte yerleştirilmelidir. Bu yüzden, periyodik grupları kapsayan alanlar diğer alanlardan sonra kayıt içinde yer alır.

Diğer yandan, ORDERS kütüğü kayıt yapısı, sipariş yönetiminde çok uygun olmasına rağmen, envanter yönetiminde uygun olmayabilir. ORDERS kütüğündeki siparişler için bir stok kontrol uygulaması tamamen farklı bir kayıt yapısı gerektirebilir. Bu kayıtlar STOCK adında farklı bir veritabanı kütüğünde tutulur (Tablo 2-2).

Tablo 2-2 Multiple-Value Alan Örneği

Multiple-value alanlar

Item_code	On_hand	Req_date	Quantity Order_item
80819W			
337015Y		1006	200
27395R	200	3105	300

STOCK kütüğündeki kayıt formatı, stok yönetimi gereken uygulamalara ORDERS kütüğündeki kayıt formatından daha uygundur. Kayıt, bir sipariş artık envanterde bulunmayan başka bir tanesinin yerine konarak tasarlandığı durumları da kontrol etmek üzere tasarlanmıştır. ITEM_CODE alanı için multiple-valuelara izin vererek, geçerli olan stok maddesi (yeni siparişin) yerini aldığı artık kullanılmayan siparişlerin numaralarıyla etiketlenebilir, bu durumda yeni değişmiş siparişi otomatik olarak seçebilmek için eski siparişlere referansa izin verilmelidir. Bunu yapabilmek için, ITEM_CODE alanı multiple-value alan olarak tanımlanmıştır.

Örneğin, 80819W ve 337015Y siparişleri artık stokta olmasın; 27395R ana siparişi için onların sipariş kodları anlamdaş hale gelir. Devam etmeyen siparişler hakkında sorgulama yapan bir uygulama programı, önce eski kod için bütün ITEM_CODE değerlerine bakar, sonra yedeği belirlemek için multiple-value alandaki ilk ITEM_CODE değerine yönlendirilir.

ITEM_CODE alanı 1'den 191'e kadar değer içerebilir. Bununla birlikte, periyodik grubun tersine, multiple-value alandaki tekil değerler, değerlerden herhangi biri silindiğinde konumsal bütünlüğü korumaz. Örneğin, yukarıda gösterilen STOCK kaydındaki 337015Y siparişi artık verilmiyorsa ve pseudocode boşluk değerine set edildiyse, 80819W otomatik olarak ITEM_CODE altındaki ikinci occurrence olur.

Aşağıdaki limitler multiple-value alanlar veya periyodik gruplar kullanıldığında uygulanır:

- Herhangi bir multiple-value alanın içerdiği değerlerin maksimum sayısı 191'dir;

- Herhangi bir periyodik grubun içerdiği occurrence'ların maksimum sayısı 191'dir;
- Bir periyodik grup başka bir periyodik grubu içeremez;
- Bir occurrence'ın sıkıştırılmış uzunluğuna bağlı olarak, kullanımları ADABAS ın desteklediği maksimum kayıt uzunluğundan daha büyük, aşırı uzun kayıt uzunluklarına sebep olabilir.

Bir periyodik grubun içerdiği descriptorlar ve periyodik bir gruptaki alanlardan oluşturulan subdescriptorlar veya superdescriptorlar mantıksal sıralı okumada kontrol amaçlı veya Find ve Sort komutlarında sıralama anahtarı olarak kullanılamazlar. Buna ek olarak, multiple-value alanlardan oluşturulan ve/veya bir periyodik grubun içerdiği bir veya daha fazla descriptorı içermeyi gerektiren arama tekniklerinde bazı özel kurallar uygulanır.

Tekil ADABAS Kütüğünde Farklı Kayıt Tipleri

Dosya sayısını azaltmanın başka bir metodu, aynı ADABAS kütüğünde iki ayrı mantıksal kayıt tipindeki veriyi saklamaktır. Örneğin, Tablo 2-3 Customer ve Order kütüklerinin nasıl birleştirilebileceğini gösterir. Bu teknik ADABAS'ın null-value suppression avantajını kullanır.

Tablo 2-3 Çoklu Kayıt Tipleri

Birleştirilmiş kütük için Alan Tanımlama Tablosu'ndaki alanlar: **Key, Record Type, Order Data, Order Item Data** Saklanan kayıtlar: **Key Type Order Data*** **Key Type* Order Item Data** * işareti suppressed null value'ları gösterir.

Sipariş parça kaydının anahtarı bu siparişteki Sipariş No artı Satır Sıra No olabilirdi. Bu teknik müşteri ve sipariş kayıt tiplerine kontrol blokları ve yüksek seviye (UI) indeks bloklarının ortak kullanımına izin vererek I/O işlemlerini azaltır. Kütük işlemi başlamadan önce daha az blok okunmak zorundadır ve buffer pool'da öteki blok tipleri için daha çok boş alan bırakılmalıdır.

Customer ve Order kayıtları, belirli bir müşterinin bütün siparişlerinin elde edilmesinde oluşması gereken bolk sayısını azaltarak, Veri Ambarında (Data Storage) birlikte gruplanabilir. Bütün siparişler müşterinin eklendiği sırada eklenirse, gereken tüm I/O işlemleri de azalacaktır. Siparişler sonradan eklenirse, başlangıçta aynı yolla gruplanmayabilirler ama daha sonra ADAORD özelliğini kullanarak sonradan gruplanabilirler.

Anahtar, müşteri ve sipariş verilerine etkin erişimi sağlamak için dikkatlice tasarlanmalıdır. Aynı müşteriye ait farklı siparişleri ayırmak için, müşteri kaydındaki anahtar genellikle öne eklenmiş null değerine sahip olacaktır. (Tablo 2-4)

Tablo 2-4 Anahtar Yapısı

A00231	000	Order header for order A00231
A00231	001	Order Item 1
A00231	002	Order Item 2
A00231	003	Order Item 3
A00232	000	Order header for order A00232
A00232	001	Order Item 1

Program anahtar öneki içeriğinden müşteri veya sipariş kaydının hangisinin gerektiğine karar verebiliyorsa, kayıt tipi alanı gereksizdir. Program, ek olan alanları okuyabilmek veya kayıt tipleriyle bağlantılı olan tüm alanları döndürmek için bir kaydı tekrar okumak zorunda kalabilir.

Fiziksel Kütüklerin Bir Mantıksal Kütüğe Bağlanması

3-byte ISN'li bir ADABAS kütüğü maksimum 16,777,215 kayıt; 4-byte ISN'li bir kütük 4,294,967,294 kayıt içerebilir. Eğer tek tipte çok geniş sayıda kayıt varsa, kayıtların çoklu fiziksel dosya üzerine yaymaya ihtiyaç duyulabilir.

Ulaşılan dosya sayısını azaltmak için ADABAS, aynı formatta kayıtları içeren çoklu fiziksel dosyanın birlikte tek bir mantıksal dosya gibi birbirine bağlanmasına olanak tanır. Bu dosya yapısı 'expanded file' (genişletilmiş dosya) olarak adlandırılır ve bunu oluşturan fiziksel dosyalar da 'companent files' (eleman dosyalar)dır. Genişletilmiş bir dosya herbiri tek mantıksal ISN bölgesi ile, 128 eleman dosya kapsayabilir. Genişletilmiş bir dosya 4,294,967,294 kaydı aşamaz.

Uygulama programının mantıksal dosyayı adreslemesine (dosyanın adresi, genişletilmiş dosyanın ana elemanının veya 'anchor' dosyasının sayısıdır) rağmen, kriter olanak belirlenmiş alandaki veriye göre, ADABAS doğru eleman dosyayı seçer. Bu alandaki verinin sadece bir eleman dosyadaki kayıtlar için de tek olan karakteristik özellikleri vardır. Bir uygulama genişletilmiş dosyayı güncellediğinde, ADABAS hangi eleman dosyanın güncelleneceğine karar vermek için, yazılacak kaydın kriter olarak belirlenen alanındaki veriye bakar. Genişletilmiş dosyadaki veriyi okurken, ADABAS doğru eleman dosyayı bulabilmek için, anahtar olarak mantıksal ISN kullanır.

Veri Çiftleri

Bazı durumlarda, detay kayıda ulaşıldığında başlık kayıttan birkaç alana gereksinim olabilir. Örneğin, bir faturanın üretiminde sipariş parçası verisinin ve ürün kaydının bir kısmı olan ürün tanımının olması gerekebilir. Faturalama programının bu bilgiyi hızlı olarak elde edebilmesinin en basit yolu sipariş maddesi verisinde ürün tanımının bir kopyasını tutmaktır. Bu, fiziksel çiftleme olarak adlandırılır, fiziksel olarak başka bir yerde -bu durumda, ürün kaydında- sunulan verinin kopyası tutulmaktadır. Fiziksel çiftleme aynı zamanda her bir detay kaydın bazı alanları başlık kayıtta periyodik grup olarak tutulduğunda da gerçekleşir.

Bir müşteri için tüm faturalarının toplamına ihtiyaç duyan bir kredi kontrol rutini düşünelim. Bu bilgi ilgili faturaları okuyup toplayarak türetilebilir, ancak bu, çok sayıda kayıda rastgele erişimi içerebilir. Bu bilgi, müşteri kaydında kalıcı olarak saklanırsa, daha hızlı şekilde elde edilebilir. Bu, mantıksal çiftleme olarak adlandırılır, kopya kayıt başka bir yerde kayıtlı değildir, ancak diğer kayıtların içeriğiyle ifade edilmiştir.

Veri Erişim Stratejileri

Descriptor'lar kullanıcı-tanımlı arama kriteri üzerine kurulmuş bir dosyadan kayıtlar seçmek ve mantıksal sıralı okuma işlemini kontrol etmek için kullanılır. Bu nedenle descriptor'ların kullanımı bir dosya için kullanılan erişim strartejisine doğrudan bağlantılıdır. Özellikle sık güncellenen her bir descriptor için ek disk boşluğu ve işlem gerekir.

Superdescriptor 20'ye kadar alanın (ya da alanın bölümleri) birleşimiyle yaratılan descriptor'dır. Superdescriptor'ı oluşturan alanların descriptor olması gerekli değildir. Superdescriptor'lar arama kriteri değerlerin bir birleşimini içerdiğinde, sıradan descriptor'ların birleşiminden daha etkilidir. Bunun nedeni ADABAS'ın istenen kayıtların listesini üretmek için, çeşitli ISN listelerini birleştirmek yerine bir Inverted Liste'ye erişimidir. Superdescriptor'lar aynı zamanda sıradan descriptor'lar gibi, dosyanın okunduğu mantıksal sırayı kontrol amaçlı da kullanılabilir.

Subdescriptor bir alanın bir kısmından türetilen descriptor'dır. Subdescriptor'ı oluşturan alanların descriptor olması gerekli değildir. Arama kriteri alfanümerik bir alanın ilk 'n' byte'ını ya da nümerik bir alanın son 'n' byte'ını kapsayan bir değerler bölgesi içeriyorsa, subdescriptor alanın sadece ilgili byte'larından tanımlanabilir. Subdescriptor kullanımı arama kriterinin bir bölge yerine bir tekil değer gibi sunulmasına olanak verir. Bu, daha etkin arama ile sonuçlanır, bunun nedeni ADABAS'ın arasonuçları birleştirmemesidir.

2.2 NATURAL Programlama Dili

NATURAL dili, ADABAS İlişkisel Veritabanı Yönetim Sistemi'ni etkin şekilde kullanan bir dördüncü kuşak programlama dilidir. Bu kısımda, NATURAL dilini oluşturan ifadeler (statements) açıklanacaktır. İfadeler NATURAL programlama dilini oluşturan 'sözcük'lerdir.

Fonksiyonlarına göre gruplandırılmış NATURAL ifadeleri:

- Veritabanı erişimi ve güncelleme
- Aritmetik işlemler ve veri taşıma operasyonları
- Döngü işlemi (execution)
- Çıktı (output) raporlarının yaratılması
- İnteraktif işlem için ekran düzenleme
- Mantıksal koşulların işlenmesi

- Program ve rutinlerin çağrılması
- Çalışma (iş) dosyalarının kontrolü
- Sonuç yönelimli (event-driven) programlama
- Çeşitli ifadeler

Veritabanı Erişimi ve Güncelleme

Tablo 2-5'deki ifadeler bir veritabanının içerdiği bilgiye ulaşmak ve bu bilgiyi işlemek için kullanılır.

Tablo 2-5 Veritabanı Erişimi ve Güncelleme

READ	Bir veritabanı dosyasındaki kayıtları fiziksel veya mantıksal sırayla okur.
FIND	Kullanıcı tarafından belirlenen kritere göre, veritabanı dosyasından kayıt seçer.
HISTOGRAM	Bir veritabanı alanındaki değerleri okur.
GET	Verilen ISN (Internal Sequence Number) veya RNO (Record Number) ile bir kayıt okur.
GET SAME	O anda işlem yapılan kaydı tekrar okur.
ACCEPT/REJECT	Kullanıcı tarafından belirlenen kritere göre, kayıtları kabul eder/reddeder.
PASSW	Şifre-korunumlu bir dosyaya erişim için şifre sağlar.
LIMIT	READ, FIND veya HISTOGRAM döngülerinin çalışma sayısını sınırlar.
STORE	Veritabanına yeni bir kayıt ekler.
UPDATE	Veritabanındaki bir kaydı günceller.
DELETE	Veritabanından bir kaydı siler.
END TRANSACTION	Mantıksal bir transaction'ın sonlandığını gösterir.
BACKOUT TRANSACTION	Kısmen tamamlanmış mantıksal bir transaction'ı geri alır.
GET TRANSACTION DATA	Önceki END TRANSACTION ifadesiyle saklanmış olan transaction verisini okur.
RETRY	Başka bir kullanıcı için bekleme statüsünde olan kaydı tekrar okumaya çalışır.
AT START OF DATA	Döngü içinde okunan bir dizi kaydın ilki işlendiğinde icra edilecek ifadeleri belirtir.
AT END OF DATA	Döngü içinde okunan bir dizi kaydın sonuncusu işlendikten sonra icra edilecek ifadeleri belirtir.

AT BREAK	Bir kontrol alanının değeri değiştiğinde (kesme işlemi) icra edilecek ifadeleri belirtir.
BEFORE BREAK PROCESSING	Kesme işlemini icra etmeden önce icra edilecek ifadeleri belirtir.
PERFORM BREAK PROCESSING	Kesme işlemini icra eder.

Aritmetik İşlemler ve Veri Taşıma Operasyonları

Tablo 2-6'daki ifadeler aritmetik işlemler ve veri taşıma operasyonları için kullanılır.

Tablo 2-6 Aritmetik İşlemler ve Veri Taşıma Operasyonları

COMPUTE	Aritmetik işlemleri icra eder veya alanlara değer atama işlemini gerçekleştirir.
ADD	İki veya daha fazla operandı toplar.
SUBTRACT	İki veya daha fazla operandı başka bir operanddan çıkarır.
MULTIPLY	İki veya daha fazla operandı çarpar.
DIVIDE	Bir operandı diğerine böler.
MOVE	Bir operandın değerini bir veya daha fazla alana atama işlemini icra eder.
MOVE ALL	Bir değerin çoklu oluşumlarını başka bir alana atama işlemini icra eder.
COMPRESS	İki veya daha fazla alanın değerini tek bir alanda birleştirir.
SEPARATE	Bir alanın içeriğini iki veya daha fazla alana ayırır.
EXAMINE	Belirli bir değeri bir alanda arar, değiştirir ve/veya hangi sıklıkta yer aldığını sayar.
RESET	Bir alanın değerini (nümerikse) sıfırlar veya (alfanümerikse) bu değere boşluk atar, veya ilk değerine set eder.

Döngü İşlemi

Tablo 2-7'deki ifadeler döngü işleminin çalışması ile ilişkilidir.

Tablo 2-7 Döngü İşlemi

REPEAT	Bir döngüyü başlatır (ve belirlenen bir koşulda bitirir).
FOR	Bir döngüyü başlatır ve döngünün işlenme sayısını kontrol eder.
ESCAPE	Bir döngüyü durdurur.

Çıktı Raporlarının Yaratılması

Tablo 2-8'deki ifadeler çıktı raporlarının yaratılması için kullanılır.

Tablo 2-8 Çıktı Raporlarının Yaratılması

FORMAT	Çıktı parametre ayarlarını belirtir.
DISPLAY	Sütun formunda çıktısı alınacak alanları belirtir.
WRITE/PRINT	Serbest formda çıktısı alınacak alanları belirtir.
WRITE TITLE	Bir raporun her bir sayfa başında yazılacak metni belirtir.
WRITE TRAILER	Bir raporun her bir sayfa sonunda yazılacak metni belirtir.
AT TOP OF PAGE	Yeni bir çıktı sayfası başladığında icra edilecek işlemi belirtir.
AT END OF PAGE	Çıktı sayfasının sonuna ulaşıldığında icra edilecek işlemi belirtir.
SKIP	Raporda bir veya daha fazla boş satır oluşturur.
EJECT	Başlıklar olmadan bir sayfa ilerletir.
NEWPAGE	Başlıklarla birlikte bir sayfa ilerletir.
DEFINE PRINTER	Bir raporu belirli bir mantıksal çıktı varış yerine tahsis eder.
CLOSE PRINTER	Bir yazıcıyı kapatır.

İnteraktif İşlem İçin Ekran Düzenleme

Tablo 2-9'daki ifadeler verinin interaktif işlenmesi amacıyla veri ekranlarını yaratmak için kullanılır.

Tablo 2-9 İnteraktif İşlem İçin Ekran Düzenleme

INPUT	Veri gösterimi/girişi için formatlı bir ekran yaratır.
REINPUT	Bir INPUT ifadesini tekrar icra eder (önceki INPUT ifadesinde hatalı veri girildiyse).
DEFINE WINDOW	Bir pencerenin boyutunu, yerini ve özelliklerini belirtir.
SET WINDOW	Bir pencereyi aktif ve inaktif hale getirir.

Mantıksal Koşulların İşlenmesi

Tablo 2-10'daki ifadeler bir NATURAL programının çalışması sırasında bulunabilen koşullar üzerine kurulu ifadelerin çalışmasını kontrol etmek için kullanılır.

Tablo 2-10 Mantıksal Koşulların İşlenmesi

IF	Mantıksal bir koşula bağlı olarak ifadeleri icra eder.
IF SELECTION	Bir alfanümerik alanlar dizisinde bir sadece bir tanesinin bir değeri içerdiğini sorgular.
DECIDE FOR	Mantıksal koşullara bağlı olarak ifadeleri icra eder.
DECIDE ON	Bir değişkenin içeriğine bağlı olarak ifadeleri icra eder.
ON ERROR	Çalışma sırasında oluşan hatalar karşısında hangi ifadelerin icra edileceğini belirler. Bu işlem yapılmadığında NATURAL programının bitirilmesiyle sonuçlanacak, bir NATURAL hata mesajı ile karşılaşılabilir.

Program ve Rutinlerin Çağrılması

Tablo 2-11'deki ifadeler program ve rutinlerin çalışması ile birlikte kullanılır.

Tablo 2-11 Program ve Rutinlerin Çağrılması

FETCH	Bir NATURAL programını çağırır.
CALLNAT	Bir NATURAL subprogramını çağırır.
PERFORM	Bir NATURAL subroutine'ini çağırır.
DEFINE SUBROUTINE	Bir NATURAL subroutine'ini tanımlar.
ESCAPE	Bir rutinin çalışmasını durdurur.
CALL	Bir NATURAL proramından NATURAL'da yazılmamış bir programı çağırır.
CALL FILE	NATURAL'da yazılmamış bir programı, ADABAS dosyası olmayan bir dosyadan bir kayıt okumak için çağırır.

Çeşitli ifadeler

Tablo 2-12'deki ifadeler çeşitli amaçlar için kullanılır.

Tablo 2-12 Çeşitli İfadeler

DEFINE DATA	Bir NATURAL program veya rutininde kullanılacak olan veri elemanlarını tanımlar.
END	Bir NATURAL program veya rutininin kaynak kodunun sonu olduğunu gösterir.
EXAMINE TRANSLATE	Bir alanın içerdiği karakterleri büyük harfe veya küçük harfe, ya da başka karakterlere çevirir.
INCLUDE	Derleme sırasında NATURAL copycode'unu içerir.
PROCESS COMMAND	Bir komut işlemcisini çağırır.
RELEASE	NATURAL çalışma alanının içeriğini siler; bir FIND ifadesiyle tutulan ISN/RNO setlerini serbert bırakır; NATURAL küresel değişkenlerini serbest bırakır.
RUN	Bir kaynak programı derler ve çalıştırır.
SET CONTROL	NATURAL programı içinden bir NATURAL terminal komutunu icra eder.
SET KEY	Terminal tuşlarına fonksiyon ataması yapar.
SETTIME	Bir *TIMD sistem değişkeni için point-in-time referansı saptar.

Kullanıcı-Tanımlı Değişkenler

Kullanıcı-tanımlı değişkenler bir programda ya da rutinde ara sonuçları saklamak amacıyla kullanılabilir.

Kullanıcı-tanımlı değişken ismi 1-32 karakter uzunluğunda olabilir. 32 karakterden uzun değişken isimleri kullanılabilir (örneğin karmaşık uygulamalarda daha uzun ve anlamlı değişken isimleri programın okunabilirliğini artırıyorsa); bununla birlikte, sadece ilk 32 karakter anlamlıdır ve bu yüzden tek olmalıdır, kalanlar NATURAL tarafından ihmal edilecektir.

Kullanıcı-tanımlı değişken ismi NATURAL için ayrılmış anahtar kelimelerden oluşmamalıdır. Bir NATURAL programı içinde, kullanıcı-tanımlı değişken ve veritabanı alanı için aynı isim kullanılmamalıdır, referans hatalarına neden olabilir.

2.3 Attachmate HLLAPI Fonksiyonları

Attachmate HLLAPI (High Level Language Application Programming Interface) fonksiyonları host bilgisayar ile haberleşme sağlayan uygulamalar yazmak için kullanılan program-callable (program içinden çağrılabilir) fonksiyonlar setidir. Attachmate HLLAPI fonksiyonları sayesinde, programlar 3270, 5250 veya asynchronous (eşzamanlı olmayan) terminal kullanan bir operatörle aynı şekilde hostla bağlantı kurabilir.

Attachmate HLLAPI fonksiyonları PC programlarının, host ve PC session arasındaki tüm etkileşimleri kontrol edebilmesine olanak tanır. Fonksiyonlar 3270, 5250 veya asynchronous terminallerin operasyonunu ve veri yapılarını simüle eder. Program bir fonksiyonu çağırdığında, Attachmate HLLAPI yazılımı veriyi yorumlar ve işlenmesi için onu presentation space (PS) tamponuna (buffer) geçirir.

Attachmate HLLAPI fonksiyonları ile, yeni PC uygulamaları yaratılabilir veya varolanlar isteğe göre değiştirilebilir (customize). Örneğin, host verisinin PC çarşaf listesine ya da PC'deki kelime-işlemci yazılıma geçirilmesi otomatik hale getirilebilir.

HLLAPI Fonksiyonlarının Tanımları

Tablo 2-13, her bir HLLAPI fonksiyonunun amacını kısaca tanımlamaktadır:

Tablo 2-13 HLLAPI Fonksiyonları

Fonksiyon	No	Amaç
Attacmate Query System	0	Kullanılan Attachmate yazılımının versiyon numarasını ve seviyesini döndürür.
Connect Presentation Space	1	HLLAPI programını belirlenen PS'e bağlar. Program PS'e bağlandıktan sonra, bu boşluk geçerli PS olur ve host bilgisayarla tüm haberleşme

		bunun üzerinden yapılır.
Disconnect Presentation Space	2	PC programının geçerli PS'le bağlantısını koparır.
Send Key	3	Geçerli PS'de imleçin (cursor) bulunduğu yere bir keystroke veya keystroke dizisi yerleştirir.
Wait	4	Geçerli PS'in durumunu test eder.
Copy Presentation Space	5	PS'in tüm içeriğini programdaki bir stringe kopyalar.
Search Presentation Space	6	Belirlenen bir stringi geçerli PS'de arar.
Query Cursor Location	7	Geçerli PS'de imleçin bulunduğu pozisyonu döndürür.
Copy Presentation Space to String	8	Geçerli PS'in içeriğinin tamamını veya bir kısmını programda tanımlanmış bir veri stringine kopyalar.
Set HLLWin Parameters	9	Varsayılan session parametrelerini değiştirmeye izin verir.
Query Sessions	10	Emulatöre o anda tanımlı session sayısını döndürür. Ayrıca herbir session için aşağıdaki bilgileri içeren 12-byte'lık bir veri stringi döndürür: • Session kısa ismi • Session uzun ismi • Host session tipi • PS'in uzunluğu
Reserve	11	Terminal operatörünün veri girişini önlemek için geçerli olan PS'i kilitler.
Release	12	Fonksiyon 11 'Reserve' ile kilitlenmiş olan o anda hosta bağlı PS'in kilidini kaldırır.
Copy Operator Information Area	13	Geçerli PS'in Operator Information Area (OIA) (Operatör Bilgi Alanı)'nın içeriğini programa döndürür.
Query Field Attribute	14	Geçerli PS'deki belirtilen alanın özelliklerini (attribute) döndürür.
Copy String to Presentation Space	15	Programdaki bir ASCII stringi geçerli PS'deki belirtilen yere kopyalar.
WSCtrl	16	Pencere ekleme, pencere odağını değiştirme, PS pencere isimlerini sorgulama gibi özelliklerin kontrolüne izin verir.
Pause	18	PC programının, bir olayın oluşmasını belirtilen süre kadar beklemesini sağlar. Zaman döngüsü (timing loop) yerine bu fonksiyonun kullanılması tavsiye edilir.

Query System	20	Kurulu Attachmate HLLAPI'nin versiyonu ve yayınlanma tarihi bilgilerini içeren 35-byte'lık veri stringi döndürür.
Reset HLLWin	21	Fonksiyon 9 'Set HLLWin Parameters' ile değiştirilen HLLAPI'yi ilk değerlerine set eder. Ayrıca host olay bildirimini (event notification) durdurur, HLLAPI'ye ayrılmış host sessionları serbest bırakır, bağlanmış host sunum sessionlarını koparır.
Query Session Status	22	Session hakkında bilgi içeren 18-byte'lık veri stringi döndürür.
Start Host Notification	23	Attachmate HLLAPI programının PS ya da OIA'nın değişip değişmediğini belirlemesine izin verir.
Query Host Update	24	Fonksiyon 23 'Start Host Notification' ile birlikte, HLLAPI programının bu fonksiyon son çağrıldığından beri, hostun PS ya da OIA'yı değiştirip değiştirmediğini belirlemesini sağlar.
Stop Host Notification	25	Host bildirimini durdurur (deactivate), Fonksiyon 24 'Query Host Update'i disable durumuna getirir. Aynı zamanda belirlenen host sessionda host olaylarının Fonksiyon 18 'Pause'i etkilemesini engeller.
Search Field	30	Geçerli PS'deki belirtilen alanda belirli bir stringin varlığı için arama yapar.
Find Field Position	31	O anda bağlı PS'de hedef alanın başlangıç pozisyonunu döndürür.
Find Field Length	32	Geçerli PS'de hedef alanın uzunluğunu döndürür.
Copy String to Field	33	Programdan bir karakter stringini geçerli PS'de belirlenen alana kopyalar.
Copy Field to String	34	Geçerli PS'de belirli bir alandaki tüm karakterleri programdaki bir veri stringine kopyalar. Veri stringini döndürmeden önce kopyalanan karakterleri ASCII'ye çevirir.
Set Cursor	40	Geçerli PS'deki imleci konuşlandırmaya izin verir.
Start Keystroke Intercept	50	PC programının terminal operatörü tarafından girilen keystroke'ları okumasına ve değerlendirmesine izin verir.

Get Key	51	PC programının bir sessiondan keystroke'ları almasına, kabul etmesine, işlemesine veya reddetmesine izin verir.
Post Intercept Status	52	Fonksiyon 51 'Get Key' ile alınan keystroke'un kabul edilip edilmediğini bildirmek için beep sesi verir.
Stop Keystroke Intercept	53	PC programının keystroke'lara müdahale etme yeteneğini engeller.
Send File	90	Hosta dosya gönderir. HLLAPI programında uygun dosya transfer SEND (gönderme) komutunu tutmaya izin verir.
Receive File	91	Hosttan dosya alır. HLLAPI programında uygun dosya transfer RECEIVE (alma) komutunu tutmaya izin verir.
Convert Position or RowCol	99	Programdan geçilen parametrelere bağlı olarak aşağıdakilerden birini icra eder: • PS pozisyonunu satır ve sütun koordinatlarına çevirir. • Satır ve sütun koordinatlarını PS pozisyonuna çevirir.
Enumarate HLLWins	A	Bu fonksiyon aktif client uygulama pencerelerinin sayısını kontrol eder.
Query HLLWin Parameters	B	Bu fonksiyon client uygulama pencere parametrelerini döndürür.
Set Message-Loop Callback	C	'Wait', 'Pause', 'GetKey' ve eşzamanlı dosya transferleri sırasında oluşan gecikmelerdeki HLLAPI mesaj-işleme döngülerinden çağrılan herhangi bir adresi client uygulamanın kaydetmesine izin verir.

2.4 Nesne-Yönelimli Programlama

Nesne-Yönelimli Programlamaya Giriş

Nesne-yönelimli problem çözümü ve nesne-yönelimli programlama, bilgisayar programlama için, yapısal programlama dilleri tarafından desteklenen alışılmış yaklaşımlardan oldukça farklı bir düşünme tarzını ve metodolojiyi simgeler. Nesne-yönelimli dillerin güçlü özellikleri bilgisayarda problem çözümünü daha insani bir aktivite yapan kavramları destekler. C++ gibi bir dille, ucuz maliyetle daha güçlü

yazılım geliştirme araçlarını sağlayan donanım yeteneğinin arttırılmasının avantajı kullanılabilir.

C++, nesne-yönelimli fonksiyonalite ile geleneksel ve etkin yapısal bir dil olan C'nin birleştirildiği melez bir dildir. C++, programcıya ve problem çözücüye, çalışma zamanı veya hafıza verimi kaybı olmadan, nesne-yönelimli yeteneğini sunar. Üretim kalite kodu sıradan bir donanım üzerinde oluşturulabilir.

Nesne-Yönelimli Programlama

Nesne-yönelimli programlama yazılım sistemleri tasarımı ve yönetimi için göreceli olarak yeni bir metottur. Başlıca hedefleri yazılımın yaygınlığını ve yeniden kullanılabilirliğini arttırarak programcının üretim kapasitesini geliştirmek ve yazılım bakımının karmaşıklığını ve maliyetini kontrol etmektir. Nesne-yönelimli programlama kullanıldığında, yazılım geliştirmenin tasarım aşaması yönetim aşamasına daha yakından bağlanır. Yazılım tasarımı ve yönetimine yaklaşımların birçoğu gibi, bu metodun potansiyeli pratikte yapılacaklara üstün gelebilir.

Nesne-yönelimli programlama bazı ana kavramlar etrafında toplanır: Soyut veri tipleri ve sınıfları (class), tip hiyerarşileri (altsınıflar), kalıtım ve polimorfizm.

Soyut veri tipleri nesne-yönelimli programlamanın ana parçasıdır. Soyut bir veri tipi, bir tip ve birleşik operasyonlar setini kapsayan bir modeldir. Bu operasyonlar temeli oluşturan tip için tanımlanmıştır ve bu tipin davranışını karakterize eder.

Çoğu nesne-yönelimli dillerde, bir sınıf tanımı temelini oluşturan tip üzerinde icra edilebilecek tüm operasyonlar için arayüz tanımlayarak, temelini oluşturan soyut veri tipinin davranışını betimler. Sınıf tanımı aynı zamanda, tipin uygulama detaylarını veya veri yapılarını belirler. Genellikle bu uygulama detaylarına sadece o sınıfın faaliyet alanı içinde erişilebilir. Bu tip 'private' tip olarak adlandırılır. Veri tipinin tamamına ya da bir kısmına sınıfın alanı dışından da erişilebiliyorsa, bu tip 'public' olarak adlandırılır.

Tip üzerinde tanımlanan operasyonlar genellikle public ya da private olarak sınıflandırılır. Public operasyonlar sınıfın alanı dışından ulaşılabilen operasyonlardır. Private operasyonlara ise sadece sınıfın alanı içinde ulaşılabilir. Nesne-yönelimli programlamada, bir sınıf için tanımlanan operasyonlar 'metotlar' (methods) olarak adlandırılır. Bu metotlar nesne-yönelimli olmayan dillerdeki prosedürler ya da fonksiyonlarla benzeşirler. Bir sınıfın public operasyonları birçok uygulama alanında kullanılmaya yeterli ise, sınıf yeniden kullanılabilir bir yazılım öğesi için temel olarak biçimlendirilebilir.

Bir 'nesne' (object) özel bir sınıfa ait olarak bildirilen bir değişkendir. Böyle bir nesne, sınıfın tanımında belirtilen verinin bütün alanlarının (private ve public) bir kopyasını içererek, tüm durumu kapsar. Bu nesne üzerindeki işlemler, sınıf tanımında belirtilen bir ya da daha fazla metot çağrılarak icra edilir. Metot çağrısı (nesneye mesaj gönderimi) tipik olarak özel bir nesnedeki saklanan veriyi değiştirir.

Her bir sınıf değişkeni veya nesnesi sınıfın bir örneğini sunar. Aynı sınıfa ait çeşitli nesneler tanımlandıysa, tipik olarak birbirlerinden farklı değer setleri içerirler.

Nesne-yönelimli bir dil genişleyebilir, programcı belirli özellikler içeren ve davranışı sınıf tanımında karakterize edilen yeni tipler yaratabilir. Bu yeni sınıflardaki nesneler programlama dilinin kendisinde bulunan önceden tanımlanmış tipler gibi kullanılabilirler.

Nesne-yönelimli paradigma, altsınıflar yoluyla tip hiyerarşileri sağlar. Altsınıf tanımı ebeveyn sınıftan bazı özellikler miras alan, ama ebeveynle paylaşılmayan bazı özel karakteristiklere gereksinen nesneler setinin davranışını karakterize eder. Altsınıfların yaratılmasına izin verilerek yazılım geliştirmenin maliyeti ve karmaşıklığı azaltılabilir.

Altsınıflar sonradan ortaya çıkan problemlerin çözümüne önderlik edebilir. Varolan yazılım öğelerini (bu öğeler için kaynak kodlarının var olduğunu varsayarsak) değiştirmek ya da yeniden yazmak yerine, ana sınıflar setinden yeni altsınıflar yaratılabilir. Bu yeni altsınıflardaki nesneler, yazılım mimarisinin altyapısını biçimlendirirler.

Nesne-Yönelimli Problem Çözümü

Nesne-yönelimli yazılım sisteminin mimarisi, sistemin temelini oluşturan tüm verinin davranışını karakterize eden sınıflar seti etrafında kurulmuştur. Her bir sınıftaki nesneler sınıfın metotları çağrılarak, bu nesnelere mesaj gönderilerek, yönetilirler. Bu mesajlar, nesneler seti üzerinde gerçekleştirilen eylemleri sunar.

Nesne-yönelimli programlama yönetim işini yapan prosedürler yerine yönetilen veri üzerine odaklanır. Veri, yazılım ayrışımı için temel oluşturur. Nesne-yönelimli yazılım tasarımının başlıca amacı, bir yazılım sisteminin temelini oluşturan veri tiplerine veya sınıf ve altsınıflara ayrışımı ve her bir ana sınıf ve altsınıfın özelliklerinin tanımıdır. Nesneler veya sınıf (altsınıf) değişkenleri, gerçek problemdeki fiziksel veya mantıksal varlıkları karşılar.

Bir nesne-yönelimli yazılım sistemini tanımlayan ve sistemin yüksek seviye tasarımını şekillendiren mimari çatı, (alt)sınıfların, tanımlarının ve nesnelerin sadece bir setini gösterir. Her bir sınıfın davranışı metot arayüzleri ile karakterize edilir. Metotların uygulama detayları sistemin yüksek seviye tasarımının bir parçası değildir.

Tipik bir nesne-yönelimli dilde, bir sınıfta tanımlı metotların arayüzü, sistem tasarımının uygulamadan ayrı tutulmasına izin verilerek, uygulama detaylarından ayrı şekilde belirlenebilir. Bir kavramın (metot arayüzleri ile sınıf tanımı) uygulamasından (veri yapısını gerçekleştiren ve sınıfın algoritmasını oluşturan kod) ayrılması nesne-yönelimli programlamada ve yeniden kullanımı ve bakım maliyetinin kontrolünü başarmada başlıca önemli noktadır.

Yeniden kullanım özelliği nesne-yönelimli programlamada artmıştır, bir sınıfın içerdiği kavramlar metot arayüzleriyle sağlanır. Kullanıcının sadece metot arayüzlerinde belirtilen sınıf nesnelerinin davranışlarını anlamaya ihtiyacı vardır, onların nasıl işletildiği ile ilgilenmek durumunda değildir. Kullanıcının bakış açısıyla, metotların nasıl uygulandığı gizli bir 'kara kutu' içindedir.

Bu yaklaşımla bakım özelliği artmıştır, veri yapısının veya algoritmanın (sınıf yönetimi içindeki kod) yönetimindeki değişimler, sınıfı ya da sınıfın bir kısmını yöneten kod bölgesine lokalize edilebilir. Sınıf dışındaki alanda bozulma etkilerine neden olmaz, sınıf arayüzü korunmaktadır. Bu arayüz, sınıfın alanı dışındaki sınıf nesneleri üzerinde gerçekleştirilen eylemler bakımından sınıfın 'kullanımı' için temeli biçimlendirir.

Nesne-yönelimli yazılım geliştirmenin başlıca hedefleri:

- Yeniden kullanılabilir yazılım öğelerini anahat sınıflar formunda kullanarak ve daha sonra ortaya çıkan problemlerin çözümünde altsınıfları görevlendirerek geliştirme süresini kısaltmak ve maliyeti düşürmek.

- Bir veya daha fazla sınıfın yönetimindeki değişimleri lokalize etme yeteneği yoluyla, yazılım bakımı maliyetini düşürmek.

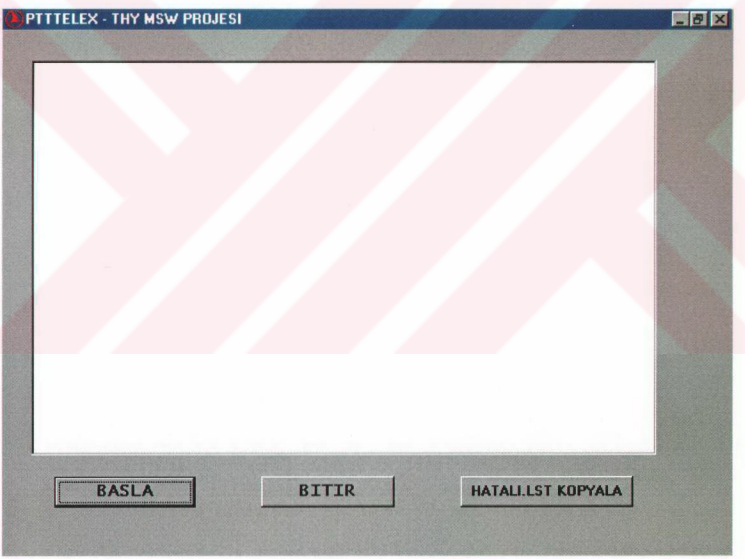
Bir nesne-yönelimli sistemin güvenilirliği arttırılabilir, bunun nedeni yüksek seviye entegrasyonun başlangıçtaki tasarımın içine kurulabilir olmasındandır. Sistemi oluşturan başlıca kısımlar, başlangıçta düzenlenmiş ve birlikte yerleştirilmişlerdir. Her bir ana kısım kendi soyut özellikleri ile tanımlanır. Yüksek seviye entegrasyon testi çoğu düşük seviyedeki detayların tamamlanmasından önce icra edilebilir. Bu, güvenilirliğin gelişmesine katkıda bulunur.

Nesne-yönelimli programlama hızlı modelleme için kullanışlı bir platform sağlar. Bir sistemin sınıflara ve bu sınıflardaki nesnelere yüksek seviyede ayrışımı tamamlandıktan sonra, sistemin parçalarının birbirine uyup uymadığını görmek için sistemin davranışını karakterize eden çoğu önemli metotlar basit ve etkisiz kodla hızlı şekilde ortaya çıkarılabilir. Daha sonra uygulama detayları geliştirilebilir.

3. UYGULAMANIN TANITIMI

3.1 Uygulamanın İşleyişi

PC'deki uygulama çalıştırılınca Şekil 3-1'deki ekran görüntülenir. Kullanıcının 'BASLA' butonuna basmasıyla C++ programı, HLLAPI fonksiyonları aracılığıyla, mainframe tarafındaki programı tetikler.



Şekil 3-1 Ptttelex Programı Ana Ekranı

Bu işlem, 'PLAY.MAC' adlı dosyanın okunup ayrıştırılmasıyla gerçekleşir. Bu dosyanın içeriği aşağıdaki gibidir:

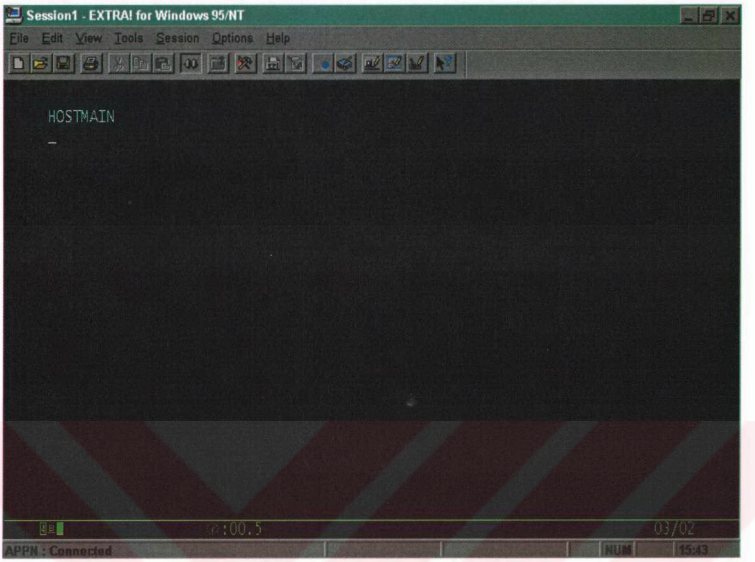
PLAY.MAC

*MSW ,PTT Telex login macrosu

```
C
(
!VTAM 'a Girilmeye Calisiliyor
<VTAM
C
WCESF LOGOFF
E
D2
<VTAM
C
WZLOGF
E
D2
)
?VTAM
!VTAM paneli geldi
WB
E
>VTAM
D8
?THY  CICSPROD
(
<    MVS PRODUCTION
? THY  CICSPROD  ORTAMI
!CICSPROD 'a girildi
C
WNAT2
E
)
WMSWLIVE MSWTLX  TELEX
E
>CICSPROD
?HOSTMAIN
T
>
F
```

Bu dosyanın içeriği istenildiği zaman değiştirilebilir. 'CSession.cpp' programındaki 'play' fonksiyonu sayesinde, yapılacak işlemlerde değişiklik olduğunda, programa müdahale edilmeksizin değişiklikler tamamlanabilir.

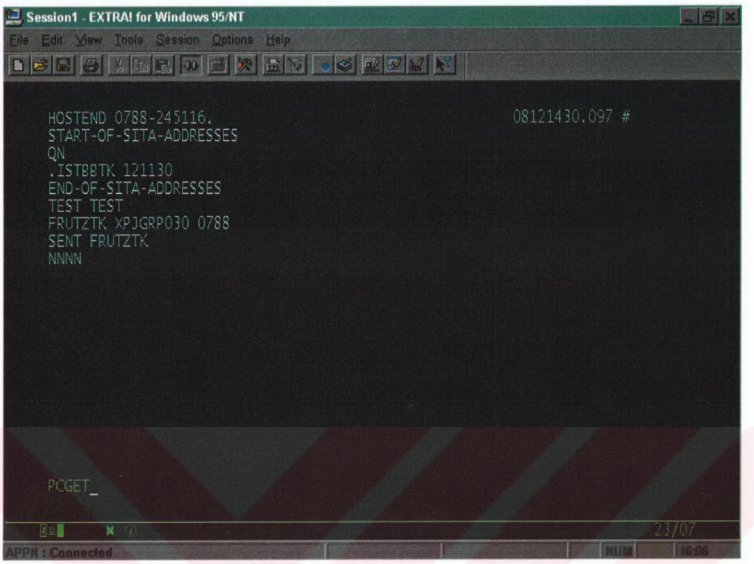
Host tarafindaki NATURAL programı PC tarafından çalıştırılınca Şekil 3-2'deki ekran görüntülenir.



Şekil 3-2 Protokol Ekranı Başlangıç Durumu

Bu ekran, hostun mesaj alışverişine hazır olduğunu göstermektedir. PC bu ekrana 'ENTER' göndererek, hosttan gelecek yanıtı bekler. Host, veritabanında PC'ye gönderilecek mesaj olup olmadığını araştırır.

Bu aşamada gönderilecek mesaj bulunuyorsa, Şekil 3-3'deki gibi bir ekranla mesajı PC'ye gönderir. Bu ekranda, 'HOSTEND' gönderilen mesajın son sayfası olduğunu gösterir. '0788-245116' bilgisi mesajın gönderileceği teleks numarasını, '08121430.097' bilgisi mesajın tarih bilgisini (MMDDHHMM.SST – ay, gün, saat, dakika, saniye, salise) belirtir. İkinci satırdan itibaren mesajın metni gönderilir. Bu bilgileri alan PC, hosta 'PCGET' anahtar kelimesini göndererek, mesajı aldığını hosta bildirir. Host bu durumda, mesajın gönderildiğini varsayarak, veritabanındaki ilgili kaydı günceller. PC ise, aldığı mesaj bilgilerini kaydeder. Mesajın tarih bilgisini mesajın dosya ismi olarak belirler, mesajın metni dosyanın içeriğini oluşturur. Gönderilecek telex numarası ile birlikte dosya ismini ve adresini 'Mesajlar.tlx' ve 'Giden.lst' kuyruk dosyalarına yazar.



Şekil 3-3 Protokol Ekranı Hosttan PC'ye Mesaj Gönderimi

Bu durumda 'Mesajlar.tlx' dosyasının içeriği aşağıdaki gibidir:

MESAJLAR.TLX

5 , 042-310093,d:\telex\gidecek\08121429.435
5 , 042-970052,d:\telex\gidecek\08121429.107
5 , 83254,d:\telex\gidecek\08101410.487
5 , 21198,d:\telex\gidecek\08031455.518
5 , 84008,d:\telex\gidecek\09171252.281
5 , 0788-245116,d:\telex\gidecek\08121430.097

Burada 5 rakamı gönderilecek mesajın önceliğini göstermektedir.

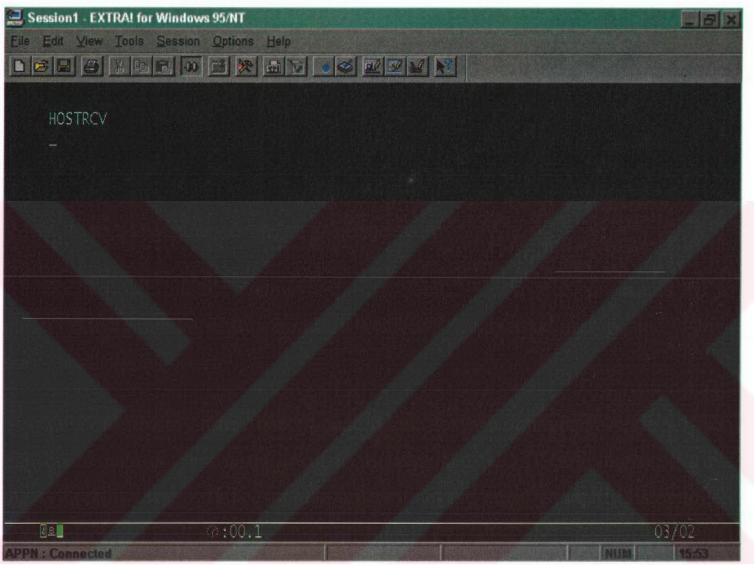
Bu durumda 'Giden.lst' dosyasının içeriği ise aşağıdaki gibidir:

GIDEN.LST

042-310093,d:\telex\gidecek\08121429.435,2
042-970052,d:\telex\gidecek\08121429.107,2
83254,d:\telex\gidecek\08101410.487,2
21198,d:\telex\gidecek\08031455.518,2
84008,d:\telex\gidecek\09171252.281,1
72223,t:\telex\gidecek\09171215.540,3
0788-245116,d:\telex\gidecek\08121430.097,0

Burada satır sonundaki rakamlar, mesajın kaç kez gönderilmek üzere 'Mesajlar.tlx' dosyasına yazıldığını göstermektedir.

Gönderilecek mesaj bulunmuyorsa host, Şekil 3-4'deki gibi bir ekranla PC'ye mesaj beklediğini bildirir. Bu ekranda, 'HOSTRCV' anahtar kelimesi hostun PC'den mesaj beklediğini gösterir.

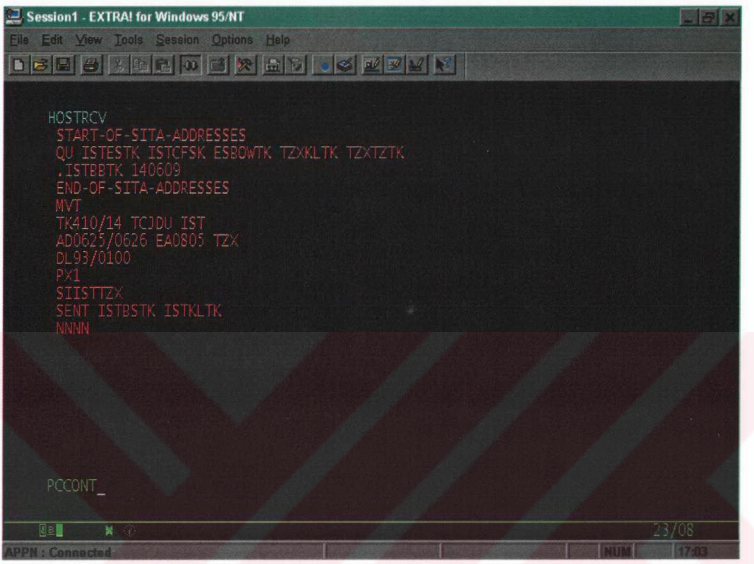


Şekil 3-4 Protokol Ekranı Hostun PC'den Mesaj Bekleme Durumu

PC bu durumda 'GELEN' dizini altında hosta gönderilecek dosya olup olmadığını kontrol eder. Dosya yoksa, 'Mesajlar.tlx', 'Giden.lst' ve 'Hatali.lst' dosyalarının içeriğini kontrol eder. Bu dosyaların son durumlarını kullanıcıya arayüz ekranında görüntüler. Bu şekilde döngüye devam eder.

'GELEN' dizini altında hosta gönderilecek dosya bulunuyorsa, Şekil 3-5'deki gibi bir ekranla dosyayı hosta gönderir. Bu ekranda, 'HOSTRCV' hostun PC'den mesaj aldığını gösterir. İkinci satırdan itibaren mesajın metni gönderilir. Bu bilgileri alan host, mesajı ilgili adreslere gönderir. Burada PC, 'PCCONT' anahtar kelimesini göndererek, mesajı hosta gönderdiğini bildirmektedir. PC hosttan yanıt alınca,

mesajın gönderildiğini varsayarak, dosyayı 'GELEN' dizini altından silerek 'GLNSAKLA' dizini altına kaydeder.



Şekil 3-5 Protokol Ekranı Hostun PC'den Mesaj Alımı

Bu aşamadan sonra, PC uygulaması sonsuz döngüsüne devam eder. Kullanıcı uygulamadan çıkmak isterse, 'CIKIS' butonuna basarak uygulama sona erdirir.

3.2 NATURAL Kaynak Kodları

Aşağıda NATURAL kaynak kodunda bulunan subroutineler açıklanmıştır.

- MAINGONDER: Protokol ekranına 'HOSTMAIN' anahtar kelimesini gönderir. Bu anahtar, PC'ye hostun ana ekranda olduğunu belirtir.
- MESAJBUL: Veritabanında PC'ye gönderilecek mesaj arar. Bulursa PC'ye gönderir. PC'den 'PCGET' yanıtı alırsa, veritabanında ilgili kaydı günceller.
- MESAJAL: PC'nin gönderdiği mesajı alır. Mainframe'de ilgili adreslere gönderir.

- PCSONGONDER: Protokol ekranına 'HOSTEND' anahtar kelimesini gönderir. Bu anahtar, PC'ye hostun mesajın son parçasında olduğunu belirtir.
- SAATGRP: PC'ye gönderilecek mesajın saat grubunu bulur.
- PCYEGONDER: Protokol ekranına 'HOSTSEND' anahtar kelimesini gönderir. Bu anahtar, PC'ye hostun mesaj gönderme ekranında olduğunu belirtir.
- PCDENAL: Protokol ekranına 'HOSTRCV' anahtar kelimesini gönderir. Bu anahtar, PC'ye hostun PC'den mesaj almak üzere beklediğini belirtir.
- MESAJGONDER: Gerekli parametreleri doldurup mesaj gönderim subprogramını çağırır.

Aşağıda NATURAL kaynak kodunda bulunan subprogramlar açıklanmıştır.

- MSWGMTFR: Verilen bir liman için, verilen bir tarihteki GMT saat farkını döndürür.
- MSWDATE1: Verilen MMDD (Ay, Gün) şeklindeki tarihe verilen gün sayısını ekler. (Gün sayısı negatif ise o tarihten çıkarır.)
- PTTAFTN1: PC'nin gönderdiği AFTN (DHMİ gönderir) mesajını ayrıştırarak, hangi adresten geldiğini belirler.
- MSW-SEND: Mesajı iletilmesi gereken adreslere gönderen subprogramdır.

Kullanılan NATURAL program ve subprogram kodları Ek A'da verilmiştir.

3.3 C++ Kaynak Kodları

Aşağıda C++ kaynak kodunda bulunan sınıflar açıklanmıştır.

- class CHllapi: HLLAPI fonksiyonlarını çağıran fonksiyonları içeren sınıftır.
- class CSession: CHllapi sınıfının altsınıfıdır. CHllapi özelliklerini kullanarak, hostla bağlantıyı oluşturan fonksiyonları içeren sınıftır.
- class TMainf: (Borland Visual C++ Builder built-in fonksiyonudur.) TForm sınıfının altsınıfıdır. Uygulama penceresindeki nesneleri ve onlar üzerine yapılan operasyonları içerir.

Aşağıda C++ kaynak kodunda bulunan fonksiyonlar açıklanmıştır.

Main.cpp dosyasında bulunan fonksiyonlar:

- controlTelex(): Manual teleks makinelerine gönderilmiş teleksleri kontrol eder, normal gönderilmişse 'GITMIS' dizini altına yerleştirir, varsa 'Hatali.lst' dosyasından siler. Hatalı gönderilmişse 'Giden.lst' yedek dosyasına tekrar yazar.
- ctrlMesajtlx(): Gitmiş mesajı 'Mesajlar.tlx' dosyasında arar, varsa siler.
- ctrlGidenlst(): Gitmiş mesajı 'Giden.lst' dosyasında arar, varsa siler. Gitmemiş mesaj bu dosyada bulunmuyorsa yeniden yazar.
- ctrlHatali(): Gitmiş mesajı 'Hatali.lst' dosyasında arar, varsa siler. Gitmemiş mesaj bu dosyada bulunmuyorsa yeniden yazar.
- checkGidenlst(): 'Giden.lst' dosyasını kontrol eder. 'Sureset.stp' dosyasında belirtilen süre kadar önce gelen ve gitmemiş mesajları 'Mesajlar.tlx' dosyasına yeniden yazar.

Host.cpp dosyasında bulunan fonksiyonlar:

- login(): Hosttaki uygulamaya login olur, NATURAL programını çağırır.
- mswHost(): Hostun gönderdiği anahtar kelimeyi okuyarak, uygulamayı mesaj gönderim ya da alım fonksiyonlarından birine yönlendirir.
- hostToPc(): Hostun gönderdiği mesajı alır, 'GIDECEK' dizini altına yazar. Mesajın teleks numarası ve dosya ismini 'Mesajlar.tlx' ve 'Giden.lst' dosyalarına yazar.
- pcToHost(): 'GELEN' dizini altında yeni mesaj olup olmadığına bakar, varsa protokol ekranına yazarak hosta gönderir. Hosta gönderilen mesajı 'GELEN' dizini altından siler, 'GLNSAKLA' dizinine yazar.
- hostWrite(char(Record[RECLEN]), int loc): Protokol ekranında 'loc' satırına Record[RECLEN] içeriğindeki bilgileri yazar.
- resetHllapi(): Hosttaki uygulamadan çıkar, session ekranını başlangıç haline getirir.

CSession.cpp dosyasında bulunan fonksiyonlar:

- initSession(): Bağlantı kurulmuş session parametrelerine başlangıç değerlerini atar.
- play(const char* MacroName): 'MacroName' dosyasını ayrıştırarak, dosyada yazılı işlemleri yapar.

- quitMsg(const char *MacroName, char *linestr): 'MacroName' ve 'linestr' içeriklerini birleştirerek, ekranı hata mesajı yazar.

CHllapi.cpp dosyasında bulunan fonksiyonlar:

- Init(): HLL_ResetHLLWin fonksiyonunu çağırır. HLLAPI parametrelerini resetler.
- Wait(): HLL_Wait fonksiyonunu çağırır, PC'nin beklemesini sağlar.
- Connect(char *PsId): HLL_ConnectPS fonksiyonunu çağırır. 'PsId' ile belirtilen session'a bağlanır.
- SearchString(LPSTR Str): HLL_SearchPS fonksiyonunu çağırır. Protokol ekranında 'Str' stringini arar.
- SendKeys(LPSTR Str): HLL_ SendKey fonksiyonunu çağırır. 'Str' içeriğini protokol ekranına gönderir.
- Clear(): SendKeys(CLEAR) fonksiyonunu çağırırarak, protokol ekranına CLEAR karakteri gönderir.
- Enter(): SendKeys(ENTER) fonksiyonunu çağırırarak, protokol ekranına ENTER karakteri gönderir.
- Pause(WORD Delay): HLL_Pause fonksiyonunu çağırır, 'Delay' kadar gecikme sağlar.
- SetCursor(int Lin): HLL_ SetCursor fonksiyonunu çağırır, protokol ekranında imleçi 'Lin' satırına konumlandırır.
- GetString(LPSTR Str, int Lin): HLL_CopyPSToString fonksiyonunu çağırır, protokol ekranında 'Lin' satırındaki stringi 'Str' stringine kopyalar. ASCII dışındaki karakterleri boşlukla değiştirir.
- Disconnect(): HLL_ DisconnectPS fonksiyonunu çağırır, session bağlantısını koparır.
- onError(int funccode, int retcode): Hata oluştuğunda, HLLAPI fonksiyon kodunu ve hata kodunu kullanıcıya bildirir.

Utility.cpp dosyasında bulunan fonksiyonlar:

- timer(clock_t delay): Uygulamanın 'delay' kadar beklemesini sağlar.
- showMesajtlx(): 'Mesajlar.tlx' dosyasının içeriğini uygulama penceresinde kullanıcıya sunar.

- showHatali(): 'Hatali.lst' dosyasının içeriğini uygulama penceresinde kullanıcıya sunar.
- crBul(char* pSatir): 'pSatir' içeriğindeki 'CR' ve 'LF' karakterlerini boşlukla değiştirir.
- finish(): HLLAPI fonksiyonlarını resetleyerek uygulamadan normal şekilde çıkışı sağlar.

Kullanılan Visual C++ program ve header kodları Ek B'de verilmiştir.

4. SONUÇ VE ÖNERİLER

Mainframe'deki bilginin kişisel bilgisayarlardaki verilere göre çok daha güvenilir olmasına rağmen, başka sistemlere taşınması zorluğu böyle bir uygulamayı gerekli hale getirmiştir.

Sonuç olarak, mainframe ve PC arasında iletişim sağlanarak, mainframe'de veri tabanındaki bilginin PC ortamına taşınması gerçekleştirilmiştir. Sistem bağlantısı olmayan noktalarda veritabanındaki güvenilir bilgiye erişim operatöre gerek olmaksızın bilgisayar yoluyla kolay bir işlem haline gelmiştir.

Bu uygulamanın gerçekleşmesinde HLLAPI fonksiyonlarının kullanımının etkisi büyüktür. PC'nin host uygulamasına bir terminal operatörü gibi girmesi ve terminal operatörünün yaptığı işlemleri yapabilmesi mainframe ve PC arasında haberleşmeyi mümkün kılmıştır.

Uygulama şu anda THY Haberleşme Merkezi'nde 24 saat kullanılmaktadır. 24 saat kullanım sonucu host bağlantısında kimi zaman oluşan problemler operatör yoluyla çözülmektedir. Uygulamaya bu tip host kaynaklı problemleri yakalayıp çözümleyecek şekilde kod eklenebilir. Ayrıca programda sonsuz döngüyü oluşturmak için kullanılan C++ fonksiyonları yerine Windows işletim sisteminde daha etkin çalışmayı sağlayan, Windows işletim sistemiyle daha yakın etkileşimi mümkün kılan Windows fonksiyonları kullanılabilir.

KAYNAKLAR

Attachmate Corporation, 1997. *Attachmate HLLAPI Programmer's Guide*, Attachmate Corporation, USA.

SOFTWARE AG Documentation Department, 1997. *ADABAS Vers. 6.2 DBA Reference Manual*, SOFTWARE AG, Germany & SOFTWARE AG Americas, Inc., Germany.

SOFTWARE AG Documentation Department, 1997. *NATURAL Vers. 2.3/3.1 Reference Manual*, SOFTWARE AG, Germany.

SOFTWARE AG Documentation Department, 1997. *NATURAL Vers. 2.3/3.1 Statements Manual*, SOFTWARE AG, Germany.

SOFTWARE AG Documentation Department, 1997. *NATURAL Vers. 2.3/3.1 Programming Guide*, SOFTWARE AG, Germany.

Wiener R. S., Pinson L. J., 1988. *An Introduction To Object-Oriented Programming and C++*, Addison-Wesley Publishing Company, Canada

EK A

PTTTELEX

```
0010 DEFINE DATA
0020 LOCAL
0030 1 MSW-FILE1-VIEW VIEW OF MSW-FILE1
0040   2 FLOG-MSG#
0050   2 FLOG-TO(1:99)
0060   2 FLOG-TIMESTAMP
0070 *
0080 1 OFFICE-TELEX-VIEW VIEW OF OFFICE-TELEX
0090   2 ZEIT-STAMP
0100   2 T-1(1:25)
0110 *
0120 1 MSW-FILE3-VIEW VIEW OF MSW-FILE3
0130   2 FTAB-PTT
0140   2 FTAB-LGADD2
0150   2 FTAB-ANSWER
0160 *
0170 1 #ILKSAYFA          (A13)
0180 1 REDEFINE #ILKSAYFA
0190   2 #ILKTIME1        (A12)
0200   2 #SONTIME1        (A1)
0210 1 #SONSAYFA          (A13)
0220 1 REDEFINE #SONSAYFA
0230   2 #ILKTIME2        (A12)
0240   2 #SONTIME2        (A1)
0250 1 #ALVERDIZI         (A69/20)
0260 1 REDEFINE #ALVERDIZI
0270   2 #YENITANIM       (20)
0280     3 #ILK3          (A3)
0290 1 REDEFINE #ALVERDIZI
0300   2 #YENITANIM1      (20)
0310     3 #ILK4          (A4)
0320 1 REDEFINE #ALVERDIZI
0330   2 #YENITANIM2      (20)
0340     3 #IKI41         (A1)
0350     3 #IKI4          (A4)
0360 1 REDEFINE #ALVERDIZI
0370   2 #YENITANIM3      (20)
0380     3 #IKI31         (A1)
0390     3 #IKI3          (A3)
0400 1 #ALVERSAY          (P3)
0410 1 #ALLINE            (A69) /* TELEX NUMARASININ BULUNDUGU SATIR
0420 1 REDEFINE #ALLINE
0430   2 #ALLDIZI         (A1/69)
0440 1 #GPTTADRES         (A8) /* PTT TELEX ADRESININ SITA ADRESI
0450 1 #PCCEVAP           (A60)
0460 1 REDEFINE #PCCEVAP
0470   2 #PCSTATUS        (A10)
0480   2 #PCINFO          (A1/50)
0490 1 #K                 (P3)
```

```
0500 1 #L                (P3)
0510 1 #LP               (P3)
0520 1 #SENDADRES        (A8) INIT<'XPJGRPTT'>
0530 1 #SENDADRES1       (A8) INIT<'XPRTI135'>
0540 1 #PTTADRESS        (A45) INIT<' '>
0550 1 #MSWADRESD        (A9)
0560 1 REDEFINE #MSWADRESD
0570   2 #MSWBOSD        (A1)
0580   2 #MSWADRESD1     (A8)
0590 1 #MSWADRESY        (A9)
0600 1 REDEFINE #MSWADRESY
0610   2 #MSWBOSY        (A1)
0620   2 #MSWADRESY1     (A8)
0630 1 #MSWADRES         (A9)
0640 1 REDEFINE #MSWADRES
0650   2 #MSWBOS         (A1)
0660   2 #MSWADRES1      (A8)
0670 1 #NOKTABUL         (A1)
0680 1 #ZAMANBAS         (A12)
0690 1 REDEFINE #ZAMANBAS
0700   2 #ZYIL           (A1)
0710   2 #ZAY            (N2)
0720   2 #SAATGRUP       (A6)
0730   2 REDEFINE #SAATGRUP
0740     3 #DD           (N2)
0750     3 #HH           (N2)
0760 1 #GMT              (N2) INIT<3>
0770 1 #TIMEST           (A12)
0780 1 REDEFINE #TIMEST
0790   2 #TM1            (A1)
0800   2 #TIMENAME       (A8)
0810   2 #TIMEEXT        (A3)
0820 1 #FILENAME         (A12)
0830 1 REDEFINE #FILENAME
0840   2 #FNAME          (A8) /* 8 KARAKTER NAME
0850   2 #POINT          (A1) /* .
0860   2 #FEXT           (A3) /* 3 KARAKTER EXTENSION DOS ICIN
0870 1 #PCTELEXNO        (A15) /* PCYE GIDECEK TELEX NOSU
0880 1 #PCTMST           (A12) /* MESAJIN TIMESTAMPI
0890 1 REDEFINE #PCTMST
0900   2 #PCTSTY         (A1)
0910   2 #PCTST          (A11)
0920 1 #PCGISN           (P11)
0930 1 #ARAR             (A1/2) INIT<'M','R'>
0940 1 #FTABPTT          (A15)
0950 1 REDEFINE #FTABPTT
0960   2 #FTABP          (A1/15)
0970 1 #FTABPTT1         (A15)
0980 1 #FP               (P3)
0990 1 #SAATGR           (A6)
1000 1 REDEFINE #SAATGR
1010   2 #SAATBR         (N6)
1020 1 REDEFINE #SAATGR
1030   2 #GYY            (A2)
1040   2 #GMM            (N2)
```

```
1050   2 #GDD            (N2)
1060 1 #ZGUNDIF          (N2) INIT<-1>
1070 1 #TLXCHAR          (P5) INIT<0>
1080 1 #TLXLIMIT         (P5) INIT<2350>
1090 * MSW-SEND PARAMETRELERI
1100 1 #PRG-ERR          (A4)
1110 1 #FROM             (A8)
1120 1 #SETFROM          (A8) INIT<'QISTPPTK'> /* PTT FROM ADRESI
1130 1 #TOO              (A8/1:99)
1140 1 #TOOCNT           (P3) INIT<0>
1150 1 #UYG              (A3) INIT<'MSW'>
1160 1 #LCNT             (P3) INIT<0>
1170 1 #LINE             (A69/1:75)
1180 1 REDEFINE #LINE
1190   2 #YENILINE       (1:75)
1200     3 #LINE4        (A4)
1210 1 #REPLY            (A1) INIT<'Y'>
1220 1 #WORK             (A36)
1230 1 #URGENCY          (A2) INIT<'QD'>
1240 END-DEFINE
1250 *
1260 BACKOUT TRANSACTION
1270 SET CONTROL 'MT'
1280 REPEAT WHILE #PCSTATUS NE 'CIK'
1290   RESET #TOO(*) #TOOCNT
1300   MOVE EDITED *DATX (EM=YYMMDD) TO #SAATGR
1310   PERFORM MAINGONDER
1320   RESET #GPTTADRES
1330   IF #PCSTATUS = 'CIK'
1340     ESCAPE ROUTINE
1350   END-IF
1360   ASSIGN #NOKTABUL = ' '
1370   PERFORM MESAJBUL
1380   PERFORM MESAJAL
1390 END-REPEAT
1400 *
1410 DEFINE SUBROUTINE MESAJBUL
1420 READ MSW-FILE3-VIEW BY FTAB-PTT = '000000000000001'
1430    THRU '999999999999999'
1440 * PCYE GONDERILECEK MESAJ ARANIYOR
1450   IF FTAB-PTT <= ' '
1460     ESCAPE TOP
1470   END-IF
1480   MOVE MSW-FILE3-VIEW.FTAB-LGADD2 TO #MSWADRES1
1490   MOVE '-' TO #MSWBOS
1500   MOVE #MSWADRES1 TO #MSWADRESD1
1510   MOVE '+' TO #MSWBOSD
1520   MOVE #MSWADRES1 TO #MSWADRESY1
1530   MOVE '+' TO #MSWBOSY
1540   MOVE FTAB-PTT TO #PTTADRESS
1550   IF FTAB-ANSWER NE ' '
1560     COMPRESS #PTTADRESS '(' FTAB-ANSWER ')' INTO #PTTADRESS
1570        LEAVING NO
1580   END-IF
1590   COMPRESS #PTTADRESS '.' INTO #PTTADRESS LEAVING NO
```

```
1600   FIND MSW-FILE1-VIEW WITH FLOG-TO = #MSWADRES
1610     RESET #ALVERDIZI(*) #ALVERSAY
1620     MOVE FLOG-TIMESTAMP TO #ILKTIME1 #ILKTIME2 #ZAMANBAS
1630     MOVE FLOG-TIMESTAMP TO #TIMEST
1640     MOVE #TIMENAME TO #FNAME
1650     MOVE #TIMEEXT TO #FEXT
1660     MOVE '.' TO #POINT
1670     MOVE #DD TO #GDD
1680     MOVE #ZAY TO #GMM
1690     CALLNAT 'MSWGMTFR' 'IST' #SAATBR #GMT
1700     SUBTRACT #GMT FROM #HH
1710     IF #HH < 0
1720       ADD 24 TO #HH
1730       CALLNAT 'MSWDATE1' #ZAY #DD #ZGUNDIF
1740     END-IF
1750     MOVE 'A' TO #SONTIME1
1760     MOVE 'Z' TO #SONTIME2
1770     READ OFFICE-TELEX-VIEW BY ZEIT-STAMP = #ILKSAYFA THRU
1780        #SONSAYFA
1790       FOR #K = 1 TO 25
1800         IF T-1(#K) = 'NNNN'
1810           ADD 1 TO #ALVERSAY
1820           MOVE T-1(#K) TO #ALVERDIZI(#ALVERSAY)
1830           PERFORM PCSONGONDER /* PCYE GONDER VE DIZIYI RESETLE
1840           RESET #ALVERDIZI(*) #ALVERSAY #NOKTABUL
1850           ESCAPE BOTTOM(1770)
1860         END-IF
1870         ADD 1 TO #ALVERSAY
1880         MOVE T-1(#K) TO #ALVERDIZI(#ALVERSAY)
1890         IF #NOKTABUL = ' '
1900           PERFORM SAATGRP
1910         END-IF
1920         IF #ALVERSAY = 20 /* 20 SATIR DA BIR PCYE GONDERIR
1930           PERFORM PCYEGONDER /* PCYE GONDER VE DIZIYI RESETLE
1940           RESET #ALVERDIZI(*) #ALVERSAY
1950         END-IF
1960       END-FOR
1970     END-READ
1980     MOVE LEFT #PCSTATUS TO #PCSTATUS
1990     IF #PCSTATUS = 'PCGET'
2000       GET MSW-FILE1-VIEW *ISN(1600)
2010       FOR #K = 1 TO 99
2020         IF FLOG-TO(#K) = #MSWADRES
2030           ASSIGN FLOG-TO(#K) = #MSWADRESY
2040         END-IF
2050       END-FOR
2060       UPDATE(2000)
2070       END TRANSACTION
2080       ESCAPE ROUTINE
2090     END-IF
2100   END-FIND
2110 END-READ
2120 END-SUBROUTINE
2130 *
2140 DEFINE SUBROUTINE MESAJAL
```

```
2150 RESET #LCNT #TLXCHAR
2160 REPEAT UNTIL #PCSTATUS = 'PCEND'
2170   PERFORM PCDENAL
2180   FOR #K = 1 TO 20
2190     IF #ALVERDIZI(#K) NE ' '
2200       MOVE #ALVERDIZI(#K) TO #ALLINE
2210       FOR #L = 69 TO 1 STEP -1
2220         IF #ALLDIZI(#L) NE ' '
2230           ESCAPE BOTTOM
2240         END-IF
2250         SUBTRACT 1 FROM #TLXCHAR
2260       END-FOR
2270       ADD 69 TO #TLXCHAR
2280       IF #TLXCHAR > #TLXLIMIT
2290         PERFORM MESAJGONDER
2300         RESET #LCNT #LINE(*) #WORK
2310         MOVE 69 TO #TLXCHAR
2320       END-IF
2330       ADD 1 TO #LCNT
2340       MOVE #ALVERDIZI(#K) TO #LINE(#LCNT)
2350       IF #LCNT > 1
2360         IF #LINE4(#LCNT - 1) = 'ZCZC'
2370           CALLNAT 'PTTAFTN1' #TOO(*) #TOOCNT #LINE(#LCNT)
2380             #GPTTADRES
2390         END-IF
2400       END-IF
2410       IF #ILK3(#K) = 'NNN' OR #ILK4(#K) = '=NNN' OR
2420          #IKI3(#K) = 'NNN' OR #IKI4(#K) = '=NNN'
2430         SUBTRACT 1 FROM #LCNT
2440         PERFORM MESAJGONDER
2450         RESET #LCNT #LINE(*) #WORK
2460       END-IF
2470     END-IF
2480     IF #LCNT > 74
2490       IF #LINE(*) NE ' '
2500         ASSIGN #LCNT = 76
2510         PERFORM MESAJGONDER
2520       END-IF
2530       RESET #LCNT #LINE(*)
2540     END-IF
2550   END-FOR
2560 END-REPEAT
2570 IF #LCNT > 0
2580   IF #LINE(*) NE ' '
2590     PERFORM MESAJGONDER
2600   END-IF
2610 END-IF
2620 RESET #LCNT
2630 END TRANSACTION
2640 END-SUBROUTINE
2650 *
2660 DEFINE SUBROUTINE PCYEGONDER
2670 INPUT 'HOSTSEND' #PTTADRESS (AD=O) ' ' #FILENAME(AD=O) '#'
2680    / '-' #ALVERDIZI(1:20) (AD=O) / '-' #PCCEVAP
2690 END-SUBROUTINE
```

```
2700 *
2710 DEFINE SUBROUTINE PCSONGONDER
2720 INPUT 'HOSTEND' #PTTADRESS (AD=O) ' ' #FILENAME(AD=O) '#'
2730    / '-' #ALVERDIZI(1:20) (AD=O) / '-' #PCCEVAP
2740 END-SUBROUTINE
2750 *
2760 DEFINE SUBROUTINE PCDENAL
2770 INPUT 'HOSTRCV' / '-' #ALVERDIZI(1:20) (AD=I) / '-' #PCCEVAP
2780 END-SUBROUTINE
2790 *
2800 DEFINE SUBROUTINE MAINGONDER
2810 INPUT 'HOSTMAIN' / '-' #ALVERDIZI(1:20) (AD=I) / '-' #PCCEVAP
2820 RESET #WORK
2830 END-SUBROUTINE
2840 *
2850 DEFINE SUBROUTINE MESAJGONDER
2860 IF #GPTTADRES = ' '
2870   MOVE #SETFROM TO #FROM
2880 ELSE
2890   MOVE #GPTTADRES TO #FROM
2900 END-IF
2910 RESET #PRG-ERR
2920 IF #TOOCNT = 0
2930   ADD 1 TO #TOOCNT
2940   MOVE #SENDADRES TO #TOO(#TOOCNT)
2950   ADD 1 TO #TOOCNT
2960   MOVE #SENDADRES1 TO #TOO(#TOOCNT)
2970 END-IF
2980 CALLNAT 'MSW-SEND' #PRG-ERR #FROM #TOO(*) #UYG #LCNT #LINE(*)
3990    #WORK #REPLY #URGENCY
3000 RESET #TOO(*) #TOOCNT
3010 END-SUBROUTINE
3020 *
3030 DEFINE SUBROUTINE SAATGRP
3040 MOVE #ALVERDIZI(#ALVERSAY) TO #ALLINE
3050 IF #ALLDIZI(1) NE '.'
3060   ESCAPE ROUTINE
3070 END-IF
3080 IF #ALLDIZI(9) = '0' THRU '9' OR #ALLDIZI(10) = '0' THRU '9'
3090   ASSIGN #NOKTABUL = '.'
3100   ESCAPE ROUTINE
3110 END-IF
3120 COMPRESS #ALVERDIZI(#ALVERSAY) #SAATGRUP INTO
3130    #ALVERDIZI(#ALVERSAY)
3140 ASSIGN #NOKTABUL = '.'
3150 END-SUBROUTINE
3160 *
3170 END
```

MSWGMTFR

```
0010 DEFINE DATA
0020 PARAMETER
0030 1 LIMAN    (A3)  /* I
0040 1 TARIH    (N6)  /* I YYMMDD
```

```
0050 1 FARK     (N2)  /* SSDD
0060 *
0070 LOCAL
0080 1 TAR-UPLIM-VIEW VIEW OF TAR-UPLIM
0090   2 L-LMTFRK
0100   2 L-LMTISR
0110   2 L-DSVFRK
0120   2 L-DSVISR
0130   2 L-DBASTAR
0140   2 L-DBITTAR
0150 END-DEFINE
0160 *
0170 FIND(1) TAR-UPLIM-VIEW WITH L-LIMAN-KODU = LIMAN
0180   IF L-LMTISR = '-' THEN
0190     L-LMTFRK := L-LMTFRK * -1
0200   END-IF
0210   IF L-DSVISR = '-' THEN
0220     L-DSVFRK := L-DSVFRK * -1
0230   END-IF
0240 *
0250   IF TARIH <= L-DBITTAR AND TARIH >= L-DBASTAR THEN
0260     FARK := (L-LMTFRK + L-DSVFRK) / 100
0270   ELSE
0280     FARK := L-LMTFRK / 100
0290   END-IF
0300 END-FIND
0310 END
```

MSWDATE1

```
0010 DEFINE DATA
0020 PARAMETER
0030 1 MM      (N2)
0040 1 DD      (N2)
0050 1 FARK    (N2)
0060 LOCAL
0070 1 DATE1   (D)
0080 1 DATEA   (A8)
0090 1 REDEFINE DATEA
0100   2 D1    (N2)
0110   2 E1    (A1)
0120   2 D2    (N2)
0130   2 E2    (A1)
0140   2 D3    (N2)
0150 END-DEFINE
0160 MOVE *DATX TO DATE1
0170 MOVE MM TO D2
0180 MOVE DD TO D3
0190 ADD FARK TO DATE1
0200 MOVE  DATE1 TO DATEA
0210 MOVE D2 TO MM
0220 MOVE D3 TO DD
0230 END
```

PTTAFTN1

```
0010 DEFINE DATA PARAMETER
0020 1 TOO          (A8/99)
0030 1 TOOCNT       (P3)
0040 1 LINES        (A69)
0050 1 REDEFINE LINES
0060   2 LINE       (A1/69)
0070 1 GPTTADRES    (A8)
0080 LOCAL
0090 1 MSW-FILE3-VIEW VIEW OF MSW-FILE3
0100   2 FTAB-PHSADD(1:64)
0110 *
0120 1 K            (P3)
0130 1 L            (P3)
0140 1 ADRES        (A8)
0150 1 REDEFINE ADRES
0160   2 LT         (A2) /* BOLGE VE ULKE KODU
0170 1 MSWADRES     (A8) INIT <'XPRTI135'>
0180 END-DEFINE
0190 RESET ADRES GPTTADRES
0200 FOR K = 4 TO 69
0210   IF LINE(K) = ' '
0220     IF LT = 'LT'
0230       FIND MSW-FILE3-VIEW WITH FTAB-LGADD2 = ADRES
0240         IF NO RECORDS
0250           IF NOT (MSWADRES = TOO(*))
0260             ADD 1 TO TOOCNT
0270             MOVE MSWADRES  TO TOO(TOOCNT)
0280           END-IF
0290         END-NOREC
0300         FOR L = 1 TO 64
0310           IF FTAB-PHSADD(L) = '        '
0320             ESCAPE BOTTOM(0230)
0330           END-IF
0340           IF NOT(FTAB-PHSADD(L) = TOO(*))
0350             ADD 1 TO TOOCNT
0360             MOVE FTAB-PHSADD(L) TO TOO(TOOCNT)
0370             ASSIGN GPTTADRES = 'QISTXXTK'
0380           END-IF
0390         END-FOR
0400       END-FIND
0410     END-IF
0420     RESET ADRES
0430   ELSE
0440     COMPRESS ADRES LINE(K) INTO ADRES LEAVING NO
0450   END-IF
0460 END-FOR
0470 END
```

EK B

MAIN.H

```
//---------------------------------------------------------------------------
#ifndef mainH
#define mainH
//---------------------------------------------------------------------------
#define PSID            "A"
#define DELAY           10
#define RECLEN          80
#define HATALIREC       97
#define CR              13
#define LF              10
#define STARTSITA       "START-OF-SITA-ADDRESSES"
#define ENDSITA         "END-OF-SITA-ADDRESSES"
#define ZCZC            "ZCZC"
#define SON             "NNNN"


#define GELENPATH       "t:\\telex\\gelen\\*.*"
#define GIDENPATH       "t:\\telex\\giden\\*.*"
#define GELENDIR        "t:\\telex\\gelen\\"
#define GIDENDIR        "t:\\telex\\giden\\"
#define GLNSKLDIR       "t:\\telex\\glnsakla\\"
#define GIDECEKDIR      "t:\\telex\\gidecek\\"
#define GITMISDIR       "t:\\telex\\gitmis\\"
#define PLAYMAC         "t:\\asa\\play.mac"
#define MESAJTLX        "t:\\asa\\mesajlar.tlx"
#define MESAJTMP        "t:\\asa\\mesajtmp.tmp"
#define GIDENLST        "t:\\asa\\giden.lst"
#define GIDENTMP        "t:\\asa\\gidentmp.tmp"
#define HATALI          "t:\\asa\\hatali.lst"
#define HATALITMP       "t:\\asa\\hatali.tmp"
#define SURESTP         "t:\\asa\\sureset.stp"

#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Menus.hpp>
#include <Stdio.h>
#include <Stdlib.h>
#include <dir.h>
#include <dos.h>
```

```
#include <time.h>
#include <Winbase.h>
#include "Chllapi.h"
#include "Csession.h"
#include "Host.h"
#include "Utility.h"

BOOL controlTelex();
BOOL ctrlMesajtlx();
BOOL ctrlGidenlst();
BOOL ctrlHatali();
BOOL checkGidenlst();

//---------------------------------------------------------------------------
class TMainf : public TForm
{
__published:   // IDE-managed Components
        TButton *Button1;
        TButton *Button2;
        TMemo *Memo1;
        void __fastcall Button1Click(TObject *Sender);

        void __fastcall EndLoop(TObject *Sender, TShiftState Shift, int X, int Y);
private:        // User declarations
public:         // User declarations
        __fastcall TMainf(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern TMainf *Mainf;

//---------------------------------------------------------------------------
#endif
```

MAIN.CPP

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "main.h"
//---------------------------------------------------------------------------
#pragma resource "*.dfm"

TMainf *Mainf;
BOOL cont = TRUE, gitmis = FALSE;
```

```
AnsiString TelexNo, FN;

//---------------------------------------------------------------------------
__fastcall TMainf::TMainf(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TMainf::Button1Click(TObject *Sender)
{
        if(login()){
                Mainf->Memo1->Lines->Add("Login tamamlandi!!!");
                while(cont){
                        if(!mswHost()){
                                Application->MessageBox("Host baglantisinda hata
                                        olustu!!!", "Sistem Hatasi", MB_OK);
                                finish();
                        }
                        if(!showMesajtlx())             finish();
                        if(!showHatali())               finish();
                        if(!controlTelex())             finish();
                        if(!checkGidenlst())            finish();
                        timer((clock_t)DELAY * CLK_TCK);
                        Memo1->Clear();
                        Memo1->Refresh();
                }
        }
        else{
                Mainf->Memo1->Lines->Add("Login basarisiz!!!");
                Mainf->Memo1->Lines->Add("Yeniden deneyin...");
        }
        finish();
}
//---------------------------------------------------------------------------

BOOL controlTelex()
{
        AnsiString lineString, Satir;
        struct ffblk f;
        int linecnt, mm, i, len;
        FILE *giden;
        BOOL startofsita, endofsita, zczc, son;
        char filename[30], record[RECLEN], telex[10], newfname[33];
```

```
lineString = "Giden dosya araniyor...." + TimeToStr(Time());
Mainf->Memo1->Lines->Add(lineString.c_str());
Mainf->Memo1->Lines->Add("                    ");
if(!findfirst(GIDENPATH, &f, FA_NORMAL)){
        linecnt=1;                  gitmis = TRUE;
        startofsita = FALSE;        endofsita = FALSE;
        zczc = FALSE;               son = FALSE;
        for(i=0; i<=10; i++)        telex[i] = '\0';
        strcpy(filename, GIDENDIR);
        strcat(filename, f.ff_name);
        if ((giden = fopen(filename,"r")) == NULL){
                Application->MessageBox("Giden dosya acilamiyor!!!",
                        "Dosya Hatasi", MB_OK);
                return FALSE;
        }
        while(fgets(record,RECLEN,giden)){
                AnsiString Satir(record);
                if(linecnt < 6){
                        mm = Satir.Pos('+');
                        if(mm){
                                if(Satir.Pos('#') || Satir.Pos(','))        continue;
                                strcpy(telex, (Satir.SubString(0,mm-1)).c_str());
                        }
                }
                else{
                        if(Satir.Pos(STARTSITA))   startofsita = TRUE;
                        if(Satir.Pos(ENDSITA))             endofsita = TRUE;
                        if(Satir.Pos(ZCZC))                zczc = TRUE;
                        if(Satir.Pos(SON))                 son = TRUE;
                }
                linecnt++;
        }
        fclose(giden);

        strcpy(newfname, GITMISDIR);
        strcat(newfname, f.ff_name);
        if(!CopyFile(filename, newfname, FALSE))        return FALSE;
        DeleteFile(filename);
        Mainf->Memo1->Lines->Add("Giden dosya kopyalandi");

        if(strcmp(telex, "        ") > 0){
                AnsiString Telex(telex);
                TelexNo = Telex;
                len = TelexNo.Length();
                if(len > 5){
```

```
                        mm = TelexNo.Length()-5;
                        TelexNo = TelexNo.SubString(0, mm-1) + '-'
                                + TelexNo.SubString(mm, 7);
                }
                FN = f.ff_name;

                if(!(startofsita && endofsita && son) && !(zczc && son)){
                        lineString = "Telex No: " + TelexNo
                                + " Filename: " + FN;
                        Mainf->Memo1->Lines->Add(lineString.c_str());
                        gitmis = FALSE;
                        DeleteFile(newfname);
                }
                ctrlGidenlst();
                ctrlHatali();
                if(gitmis)              ctrlMesajtlx();
        }
    }
    return TRUE;
}

BOOL ctrlMesajtlx()
{
    FILE *mesajtlx, *mesajtmp;
    AnsiString Satir;
    char record[RECLEN];

    if ((mesajtlx = fopen(MESAJTLX,"r")) == NULL){
        Application->MessageBox("Mesajlar.tlx acilamiyor!!!",
            "Dosya Hatasi", MB_OK);
        return FALSE;
    }
    if ((mesajtmp = fopen(MESAJTMP,"wt")) == NULL){
        Application->MessageBox("Mesajlar.tlx acilamiyor!!!",
            "Dosya Hatasi", MB_OK);
        return FALSE;
    }

    while(fgets(record, RECLEN, mesajtlx)){
        AnsiString lineString(record);
        if(!(lineString.Pos(TelexNo) && lineString.Pos(FN)))
            fwrite(record, strlen(record), 1, mesajtmp);
    }
    fclose(mesajtlx);
    fclose(mesajtmp);
```

```
        if(!CopyFile(MESAJTMP, MESAJTLX, FALSE)) return FALSE;
        DeleteFile(MESAJTMP);
        Mainf->Memo1->Lines->Add("Mesajlar.tlx kopyalandi");
        return TRUE;
}

BOOL ctrlGidenlst()
{
        FILE *gidenlst, *gidentmp;
        AnsiString Satir;
        char record[RECLEN];
        BOOL present = FALSE;

        if ((gidenlst = fopen(GIDENLST,"r")) == NULL){
                Application->MessageBox("Giden.lst acilamiyor!!!",
                        "Dosya Hatasi", MB_OK);
                return FALSE;
        }
        if ((gidentmp = fopen(GIDENTMP,"wt")) == NULL){
                Application->MessageBox("Giden.lst acilamiyor!!!",
                        "Dosya Hatasi", MB_OK);
                return FALSE;
        }

        while(fgets(record, RECLEN, gidenlst)){
                AnsiString lineString(record);
                if(!(gitmis && lineString.Pos(TelexNo) && lineString.Pos(FN)))
                        fwrite(record, strlen(record), 1, gidentmp);
                if(!gitmis)
                        if(lineString.Pos(TelexNo) && lineString.Pos(FN))
                                present = TRUE;
        }
        if(!gitmis){
                Satir = " " + TelexNo + "," + GIDECEKDIR + FN + ",0";
                Satir += '\n';
                if(!present)
                        fwrite(Satir.c_str(), strlen(Satir.c_str()), 1, gidentmp);
        }
        fclose(gidenlst);
        fclose(gidentmp);
        if(!CopyFile(GIDENTMP, GIDENLST, FALSE))  return FALSE;
        DeleteFile(GIDENTMP);
        Mainf->Memo1->Lines->Add("Giden.lst kopyalandi");
        return TRUE;
}
```

```
BOOL ctrlHatali()
{
        FILE *hatali, *hatalitmp;
        char record[HATALIREC];

        if ((hatali = fopen(HATALI,"r")) == NULL){
                Application->MessageBox("Hatali.lst acilamiyor!!!",
                        "Dosya Hatasi", MB_OK);
                return FALSE;
        }
        if ((hatalitmp = fopen(HATALITMP,"wt")) == NULL){
                Application->MessageBox("Hatali.lst acilamiyor!!!",
                        "Dosya Hatasi", MB_OK);
                return FALSE;
        }

        while(fgets(record, HATALIREC, hatali)){
                AnsiString lineString(record);
                if(!(lineString.Pos(TelexNo) && lineString.Pos(FN)))
                        fwrite(record, strlen(record), 1, hatalitmp);
        }
        fclose(hatali);
        fclose(hatalitmp);
        if(!CopyFile(HATALITMP, HATALI, FALSE))    return FALSE;
        DeleteFile(HATALITMP);
        Mainf->Memo1->Lines->Add("Hatali.lst kopyalandi");
        return TRUE;
}

BOOL checkGidenlst(void)
{
        FILE *gidenlst, *gidentmp, *mesajtlx, *mesajtmp, *surestp;
        AnsiString Satir, Saat, Dakika, TlxNo, Fn;
        AnsiString Counter,ToCount, St1, Dk1, Dat1, Dat2, Datf;

        char record[RECLEN], recordm[RECLEN], recordg[RECLEN], saat[9];
        char cnt[3], sure[5] , saat[9], date[9];

        int surec, mm, mn, len, sure;
        int flag = 1, bekliyor = 0;
        int toplDakBugun, toplDakFn, dk11, dk22, tarih, tarihf, counter;
        div_t saatfark;

        if ((surestp = fopen(SURESTP,"r")) == NULL){
```

```
        Application->MessageBox("Sureset.stp dosyasi acilamiyor!!!",
                "Dosya Hatasi", MB_OK);
        return FALSE;
}
fgets(sure,RECLEN,surestp);
fclose(surestp);
surec = atoi(Ssure);

if ((gidenlst = fopen(GIDENLST,"r")) == NULL){
        Application->MessageBox("Giden.lst acilamiyor!!!",
                "Dosya Hatasi", MB_OK);
        return FALSE;
}
if ((gidentmp = fopen(GIDENTMP,"wt")) == NULL){
        Application->MessageBox("Giden.lst acilamiyor!!!",
                "Dosya Hatasi", MB_OK);
        return FALSE;
}

while(fgets(recordg, RECLEN, gidenlst)){
        AnsiString Recg(recordg);
        if(!Recg.LastDelimiter("0123456789"))      continue;
        sure = atoi(Ssure);
        ToCount = Recg;

        mm = Recg.Pos(",");
        TlxNo = Recg.SubString(2, mm-2);
        mn = Recg.Pos('.');
        Fn = Recg.SubString(mn-8, 12);
        Saat = Recg.SubString(mn-4, 2);
        Dakika = Recg.SubString(mn-2, 2);
        Datf = Recg.SubString(mn-8, 4);
        tarihf = atoi(Datf.c_str());

        AnsiString Counter(recordg);
        mm = Counter.LastDelimiter(',');
        len = Counter.Length();
        Counter = Counter.SubString(mm+1, len-mm-1);
        counter = atoi(Counter.c_str());

        _strdate(date);
        _strtime(saat);
        AnsiString CDate(date);
        Dat1 = CDate.SubString(1, 2);
        Dat2 = CDate.SubString(4, 2);
```

```
Dat1 = Dat1 + Dat2;
tarih = atoi(Dat1.c_str());
AnsiString CSaat(saat);
St1 = CSaat.SubString(1, 2);
Dk1 = CSaat.SubString(4, 2);

toplDakBugun = atoi(St1.c_str()) * 60 + atoi(Dk1.c_str());
toplDakFn = atoi(Saat.c_str()) * 60 + atoi(Dakika.c_str());

if ((mesajtlx = fopen(MESAJTLX,"r")) == NULL){
        Application->MessageBox("Mesajlar.tlx acilamiyor!!!",
                "Dosya Hatasi", MB_OK);
        return FALSE;
}

bekliyor = 0;
while(fgets(recordm, RECLEN, mesajtlx)){
        AnsiString Recm(recordm);
        if(!Recm.LastDelimiter("0123456789"))    continue;
        if(Recm.Pos(TlxNo) && Recm.Pos(Fn))    bekliyor = 1;
}
fclose(mesajtlx);

if((tarih > tarihf) && (toplDakBugun < toplDakFn))  toplDakFn = 1;

if((toplDakBugun > toplDakFn + sure) && (toplDakFn != 0)
                && (bekliyor == 0)){
        saatfark = div((toplDakBugun - toplDakFn),surec);
        counter = saatfark.quot;
        mn = ToCount.LastDelimiter(",");
        ToCount = ToCount.SubString(1, mn);
        itoa(counter,cnt,10);
        ToCount = ToCount + AnsiString(cnt);
        ToCount += '\n';
        fwrite(ToCount.c_str(), strlen(ToCount.c_str()), 1, gidentmp);

        if ((mesajtmp = fopen(MESAJTMP,"wt")) == NULL){
                Application->MessageBox(
                        "Mesajlar.tlx acilamiyor!!!",
                        "Dosya Hatasi", MB_OK);
                return FALSE;
        }
        Satir = "5  , " + TlxNo + GIDECEKDIR + Fn;
        Satir += '\n';
        fwrite(Satir.c_str(), strlen(Satir.c_str()), 1, mesajtmp);
```

```
                    if ((mesajtlx = fopen(MESAJTLX,"r")) == NULL){
                            Application->MessageBox(
                                    "Mesajlar.tlx acilamiyor!!!",
                                    "Dosya Hatasi", MB_OK);
                            return FALSE;
                    }
                    while(fgets(record, RECLEN, mesajtlx)){
                            AnsiString cnfTEMP(record);
                            if(!cnfTEMP.LastDelimiter("0123456789"))
                                    continue;
                            fwrite(record, strlen(record), 1, mesajtmp);
                    }
                    fclose(mesajtlx);
                    fclose(mesajtmp);
                    if(!CopyFile(MESAJTMP, MESAJTLX, FALSE))
                            return FALSE;
                    DeleteFile(MESAJTMP);
                    Mainf->Memo1->Lines->Add("Mesajlar.tlx kopyalandi");
                    flag = 0;
            }
            if((flag == 1) && (toplDakFn != 0))
                    fwrite(recordg, strlen(recordg), 1, gidentmp);
            flag = 1;
    }
    fclose(gidenlst);
    fclose(gidentmp);
    if(!CopyFile(GIDENTMP, GIDENLST, FALSE))  return FALSE;
    DeleteFile(GIDENTMP);
    Mainf->Memo1->Lines->Add("Giden.lst kopyalandi");
    return TRUE;
}

void __fastcall TMainf::EndLoop(TObject *Sender, TShiftState Shift, int X,
      int Y)
{
    int button;
    if(!cont)                               finish();
    button = Application->MessageBox("Cikis olsun mu?",
            "Cikis", MB_YESNO);
    if(button == IDYES)         cont = FALSE;
}
//---------------------------------------------------------------------------
```

```
void __fastcall TMainf::Button3Click(TObject *Sender)
{
        Application->MessageBox(
                "Hatali.lst deki mesajlar Mesajlar.tlx'e kopyalanacak...",
                "Dosya Transferi", MB_OK);
        if(temizleHatali())
                Application->MessageBox(
                        "Hatali.lst Mesajlar.tlx'e basariyla aktarildi...",
                        "Dosya Transferi", MB_OK);
        else
                Application->MessageBox("Hatali.lst Mesajlar.tlx'e aktarilamadi...",
                        "Hata Olustu", MB_OK);
}
//---------------------------------------------------------------------------
```

HOST.H

```
//---------------------------------------------------------------------------
#ifndef HostH
#define HostH

BOOL login();
BOOL mswHost();
BOOL hostToPc();
BOOL pcToHost();
void  hostWrite(char(Record[RECLEN]), int loc);
BOOL resetHllapi();

//---------------------------------------------------------------------------
#endif
```

HOST.CPP

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Main.h"

//---------------------------------------------------------------------------
#pragma package(smart_init)

CSession Host;
AnsiString Telex, Fn;
```

```
char proStr[COLLEN];
int linenum;

BOOL login()
{
        Host.initSession();
        return Host.play(PLAYMAC);
}

BOOL mswHost()
{
        int i,j;
        char protStr[COLLEN];

        Host.Enter();
        Host.Wait();
        Host.GetString(protStr, 2);
        Telex = '\0';
        if(!Host.SearchString("HOSTRCV")){
                Mainf->Memo1->Lines->Add("HOSTRCV");
                return (pcToHost());
        }
        else
                if(!Host.SearchString("HOSTSEND")
                                || !Host.SearchString("HOSTEND")){
                        Mainf->Memo1->Lines->Add("HOSTSEND");
                        for(i=9; (i<79 && protStr[i] != '.'); i++)
                                if(protStr[i] != ' ')
                                        Telex = Telex + protStr[i];
                        Fn = GIDECEKDIR;
                        for(j=i+1; (j<79 && protStr[j] != '#'); j++)
                                if(protStr[j] != ' ')       Fn = Fn + protStr[j];
                        return (hostToPc());
                }
                else{
                        Mainf->Memo1->Lines->Add(
                                "HOSTSEND veya HOSTRCV bulunamadi...");
                        return FALSE;
                }
}

BOOL hostToPc()
{
        char record[RECLEN], protStr[COLLEN];
        FILE *gidecek, *mesajtlx, *mesajtmp, *gidenlst, *gidentmp;
```

```
BOOL first = TRUE, present = FALSE;
AnsiString lineString, sonsatir, TlxNoRight5, songiden;
int i,len;

if ((gidecek = fopen(Fn.c_str(),"r")) != NULL){
        lineString = Fn + " tekrar gonderildi, eskisi korunuyor";
        Mainf->Memo1->Lines->Add(lineString.c_str());
        first = FALSE;
        fclose(gidecek);
}
else
        gidecek = fopen(Fn.c_str(),"at");
while(!Host.SearchString("HOSTSEND")
                || !Host.SearchString("HOSTEND")){
        for(linenum=3; linenum<23; linenum++){
                Host.GetString(protStr, linenum);
                for(i=COLLEN-1; i>=0; i--)
                        if(protStr[i] != ' '){
                                protStr[i+1] = NULL;
                                break;
                        }
                len = i+1;
                if(protStr[len] != NULL)      protStr[0] = NULL;
                lineString = " <<<< " + AnsiString(protStr);
                Mainf->Memo1->Lines->Add(lineString.c_str());
                protStr[len] = '\n';
                if(!Host.SearchString("HOSTEND") ||
                        (protStr != "                                                    "))
                        if(first)fwrite(protStr, len+1, 1, gidecek);
        }
        if(!Host.SearchString("HOSTEND")){
                if(first)       fclose(gidecek);
                if ((mesajtlx = fopen(MESAJTLX,"r")) == NULL){
                        Application->MessageBox(
                                "Mesajlar.tlx acilamiyor!!!",
                                "Dosya Hatasi", MB_OK);
                        return FALSE;
                }
                if ((mesajtmp = fopen(MESAJTMP,"wt")) == NULL){
                        Application->MessageBox(
                                "Mesajlar.tlx acilamiyor!!!",
                                "Dosya Hatasi", MB_OK);
                        return FALSE;
                }
```

```
TlxNoRight5 = Telex.SubString(Telex.Length()-4, 6);
sonsatir = Telex + "(" + TlxNoRight5 + ")," + Fn;
songiden = " " + Telex + "," + Fn;

while(fgets(record, RECLEN, mesajtlx)){
        AnsiString lineString(record);
        if(lineString.Pos(sonsatir))            present = TRUE;
        else    fwrite(record, strlen(record), 1, mesajtmp);
}

sonsatir = "5  , " + sonsatir;
sonsatir += '\n';
if(!present)
        fwrite(sonsatir.c_str(), strlen(sonsatir.c_str()),
                1, mesajtmp);
fclose(mesajtlx);
fclose(mesajtmp);
if(!CopyFile(MESAJTMP, MESAJTLX, FALSE)){
        Mainf->Memo1->Lines->Add(
        "Gelen telex mesajlar.tlx dosyasina kopyalanamadi...");
        return FALSE;
}
DeleteFile(MESAJTMP);

if ((gidenlst = fopen(GIDENLST,"r")) == NULL){
        Application->MessageBox("Giden.lst acilamiyor!!!",
                "Dosya Hatasi", MB_OK);
        return FALSE;
}
if ((gidentmp = fopen(GIDENTMP,"wt")) == NULL){
        Application->MessageBox("Giden.lst acilamiyor!!!",
                "Dosya Hatasi", MB_OK);
        return FALSE;
}

while(fgets(record, RECLEN, gidenlst)){
        AnsiString lineString(record);
        if(lineString.Pos(songiden))            present = TRUE;
        else    fwrite(record, strlen(record), 1, gidentmp);
}

songiden = songiden + ",0";
songiden += '\n';
if(!present)
        fwrite(songiden.c_str(), strlen(songiden.c_str()),
```

```
                                1, gidentmp);
                        fclose(gidenlst);
                        fclose(gidentmp);
                        if(!CopyFile(GIDENTMP, GIDENLST, FALSE)){
                                Mainf->Memo1->Lines->Add(
                                "Gelen telex giden.lst dosyasina kopyalanamadi...");
                                return FALSE;
                        }
                        DeleteFile(GIDENTMP);

                        hostWrite("PCGET", 23);
                        Host.Enter();
                        Host.Wait();
                        return TRUE;
                }
                Host.Enter();
                Host.Wait();
        }
        Mainf->Memo1->Lines->Add("HOSTSEND - HOSTEND bulunamadi...");
        return FALSE;
}

BOOL pcToHost()
{
        char filename[30], record[RECLEN], newfname[33];
        char datebuf[9], timebuf[9];
        struct ffblk f;
        FILE *gelen;
        AnsiString lineString, timeStamp;

        if(!findfirst(GELENPATH, &f, FA_NORMAL)){
                strcpy(filename, GELENDIR);
                strcat(filename, f.ff_name);
                if ((gelen = fopen(filename,"r")) == NULL){
                        Application->MessageBox("Gelen dosya acilamiyor!!!",
                                "Dosya Hatasi", MB_OK);
                        return FALSE;
                }
                linenum = 3;
                while(fgets(record,RECLEN,gelen)){
                        hostWrite(crBul(record), linenum++);
                        lineString = " > > " + AnsiString(record);
                        Mainf->Memo1->Lines->Add(lineString.c_str());
                        if(linenum > 22){
                                hostWrite("PCCONT", linenum);
```

```
                        Host.Enter();
                        Host.Wait();
                        linenum = 3;
                        if(!Host.SearchString("HOSTRCV") &&
                                !Host.SearchString("HOSTMAIN")){
                                Mainf->Memo1->Lines->Add(
                                "HOSTMAIN - HOSTRCV bulunamadi...");
                                return FALSE;
                        }
                }
        }
        hostWrite("PCEND", 23);
        Host.Enter();
        Host.Wait();
        fclose(gelen);
        if(!Host.SearchString("HOSTMAIN")){
                _strdate(datebuf);
                _strtime(timebuf);
                AnsiString SDate(datebuf);
                AnsiString STime(timebuf);
                SDate = SDate.SubString(0,2) + SDate.SubString(4,2);
                STime = STime.SubString(0,2) + STime.SubString(4,2) + "."
                                        + STime.SubString(7,2) + "0";
                timeStamp = SDate + STime;
                strcpy(newfname, GLNSKLDIR);
                strcat(newfname, timeStamp.c_str());
                if(!CopyFile(filename, newfname, FALSE)){
                        Mainf->Memo1->Lines->Add(
                                "Gelen dosya kopyalanamadi...");
                        return FALSE;
                }
                DeleteFile(filename);
                return TRUE;
        }
        else
                Mainf->Memo1->Lines->Add(
                        "Kopyalama islemi gerceklesmedi.");
    }
    else{
        hostWrite("PCEND", 23);
        Host.Enter();
        Host.Wait();
    }
    return TRUE;
}
```

```
void  hostWrite(char(Record[RECLEN]), int Line)
{
        Host.SetCursor(Line);
        Host.SendKeys(Record);
}

BOOL resetHllapi()
{
        int i=0;
        while(Host.SearchString("VTAM")){
                if(i > 5)         return FALSE;
                Host.Clear();
                Host.Wait();
                Host.Wait();
                Host.SendKeys("ZLOGF");
                Host.Enter();
                Host.Wait();
                Host.Wait();
                i++;
        }
        return TRUE;
}
```

CSESSION.H

```
#ifndef CsessionH
#define CsessionH

class CSession : public CHllapi
{
   public:
        CSession();
        void initSession();
        BOOL play(const char* MacroName);
        BOOL quitMsg(const char *MacroName, char *linestr);

   private:
};

//---------------------------------------------------------------------------
#endif
```

CSESSION.CPP

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Main.h"

//---------------------------------------------------------------------------

CSession :: CSession() : CHllapi()
{ }

void CSession :: initSession()
{
        Init();
        Connect(PSID);
}

BOOL CSession :: play(const char* MacroName)
{
        char Record[RECLEN], LineString[50], loopcnt;
        Cardinal LoopBegin = NULL, LoopEnd = NULL;
        BOOL Enable = TRUE, LoopTrue = FALSE;
        int x, delay;
        FILE *macrofp;

        if ((macrofp = fopen(MacroName,"r")) == NULL){
                Application->MessageBox("Macro dosyasi acilamiyor!!!",
                        "Dosya Hatasi", MB_OK);
                return FALSE;
        }
        while(fgets(Record,RECLEN,macrofp)){
                for(x=0; Record[x]; x++){
                        if(Record[x] == CR || Record[x] == LF)
                                Record[x] = NULL;
                        if(Record[x] == '_')            Record[x]=' ';
                }
                if(*Record == '*')              continue;
                if(Enable && (!LoopTrue || *Record == ')'))
                        switch(*Record){
                                case '(' :
                                        LoopBegin = ftell(macrofp);
                                        LoopTrue = FALSE;
                                        loopcnt = 0;
                                        break;
                                case '!' :
```

```
        Mainf->Memo1->Lines->Add(Record+1);
        break;
case '<' :
        if(!LoopBegin){
                if(!quitMsg(MacroName, Record)){
                        fclose(macrofp);
                        return FALSE;
                }
                else{
                        fseek(macrofp, 0L, SEEK_SET);
                        continue;
                }
        }
        else{
                loopcnt++;
                Wait();
                if(!SearchString(Record+1))
                        if(!LoopEnd)  LoopTrue=TRUE;
                        else{
                fseek(macrofp, LoopEnd, SEEK_SET);
                                LoopEnd = NULL;
                                LoopBegin = NULL;
                }
                else
                        if(loopcnt > 20){
                if(!quitMsg(MacroName, Record)){
                                        fclose(macrofp);
                                        return FALSE;
                                }
                                else{
                        fseek(macrofp, 0L, SEEK_SET);
                                        continue;
                                }
                        }
        }
        break;
case 'C' :
        Clear();
        Wait();
        break;
case 'W' :
        SendKeys(Record+1);
        Wait();
        break;
case 'E' :
```

```
                Enter();
                Wait();
                break;
        case ')' :
                if(!LoopBegin)
                        if(!quitMsg(MacroName, Record)){
                                fclose(macrofp);
                                return FALSE;
                        }
                        else{
                                fseek(macrofp, 0L, SEEK_SET);
                                continue;
                        }
                LoopEnd = ftell(macrofp);
                if(LoopTrue){
                        LoopEnd = NULL;
                        LoopBegin = NULL;
                        LoopTrue = FALSE;
                }
                else
                fseek(macrofp, LoopBegin, SEEK_SET);
                break;
        case '?' :
                Wait();
                if(!SearchString(Record+1)){
                        Enable=TRUE;
                        break;
                }
                else{
                        Wait();
                if(!SearchString("GIRIS YAPILAMADI")){
                        Application->MessageBox(
"Sistemde sorun var, lutfen bir sure sonra tekrar deneyiniz...",
                                "Sistem Sorunu", MB_OK);
                        fclose(macrofp);
                        return FALSE;
                        }
                        else{
                        if(!quitMsg(MacroName, Record)){
                                        fclose(macrofp);
                                        return FALSE;
                                }
                                else{
                                fseek(macrofp, 0L, SEEK_SET);
                                        continue;
```

```
                                        }
                                    }
                                }
                            case 'D' :
                                strcpy(LineString, Record+1);
                                strcat(LineString, " sn. bekliyor");
                                Mainf->Memo1->Lines->Add(LineString);
                                delay = atoi(Record+1);
                                Pause(LOWORD(delay));
                                break;
                            case 'T' :
                                fclose(macrofp);
                                return TRUE;
                        }
        }
        fclose(macrofp);
        return TRUE;
}

BOOL CSession :: quitMsg(const char *MacroName, char *linestr)
{
        char BoxString[60];
        int button;
        strcpy(BoxString, MacroName);
        strcat(BoxString, " dosyasinda '");
        strcat(BoxString, linestr);
        strcat(BoxString, "' satirinda hata!!!");
        strcat(BoxString, " Dosyayi kontrol edip tekrar deneyiniz...");
        button = Application->MessageBox(BoxString, "Dosya Hatasi",
                MB_OKCANCEL);
        if (button == IDOK)             return TRUE;
        else                            return FALSE;
}
```

CHLLAPI.H

```
//---------------------------------------------------------------------------
#ifndef ChllapiH
#define ChllapiH

#define CLEAR "@C"
#define ENTER "@E"
#define COLLEN 80

class CHllapi
```

```
{
  public:
        CHllapi();
        int Init();
        int Wait();
        int Connect(char *PsId);
        int SearchString(LPSTR Str);
        int SendKeys(LPSTR Str);
        int Clear();
        int Enter();
        int Pause(WORD Delay);
        int SetCursor(int Lin);
        int GetString(LPSTR Str, int Lin);
        int Disconnect();
        TEdit *txt;

  private:
        BOOL onError(int funccode, int retcode);
        WORD str_len;
};
//---------------------------------------------------------------------------
#endif // ChllapiH
```

CHLLAPI.CPP

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Hllapi.h"
#include "Main.h"

//---------------------------------------------------------------------------

WORD WINAPI ret_code;
HWND hWnd;
BOOL error_code;

CHllapi :: CHllapi()
{ }

CHllapi :: Init()
{
        ret_code = HLL_ResetHLLWin(hWnd);
        if(ret_code)    return(onError(HLL_RESETHLLWIN,ret_code));
```

```
        Wait();
        return HLL_SUCCESS;
}

CHllapi :: Wait()
{
        int y;
        for(y=0; y<10; y++)   ret_code = HLL_Wait(hWnd);
        while (ret_code == HLL_TIMEOUT ){
                for(y=0; y<10; y++)   HLL_Wait(hWnd);
                break;
        }
        return(0);
}

CHllapi :: Connect(char *PsId)
{
        ret_code = HLL_ConnectPS(hWnd, *PsId);
        if(ret_code)    return(onError(HLL_CONNECTPS,ret_code));
        Wait();
        return HLL_SUCCESS;
}

CHllapi :: SearchString(LPSTR Str)
{
        str_len = StrLen(Str);
        ret_code = HLL_SearchPS(hWnd,Str,str_len,HLLWIN_SRCHALL);
        if(ret_code)    return(onError(HLL_SEARCHPS,ret_code));
        Wait();
        return HLL_SUCCESS;
}

CHllapi :: SendKeys(LPSTR Str)
{
        ret_code = HLL_SendKey(hWnd,Str);
        if(ret_code)    return(onError(HLL_SENDKEY,ret_code));
        Wait();
        return HLL_SUCCESS;
}

CHllapi :: Clear()
{
        ret_code = SendKeys(CLEAR);
        return(ret_code);
```

```
        Wait();
}

CHllapi :: Enter()
{
        ret_code = SendKeys(ENTER);
        return(ret_code);
        Wait();
}

CHllapi :: Pause(WORD Delay)
{
        ret_code = HLL_Pause(hWnd, Delay);
        if(ret_code)    return(onError(HLL_PAUSE,ret_code));
        Wait();
        return HLL_SUCCESS;
}

CHllapi :: GetString(LPSTR Str, int Lin)
{
        WORD length = COLLEN;
        int i;
        Lin--;
        WORD Pos = LOWORD(Lin*80+1);
        ret_code = HLL_CopyPSToString(hWnd, Str, length, Pos);
        if(ret_code)    return(onError(HLL_COPYPSTOSTRING,ret_code));

        for(i=0; i<COLLEN; i++)
                if(Str[i] < 32 || Str[i] > 126)                 Str[i] = ' ';
        Str[length] = NULL;
        return HLL_SUCCESS;
}

CHllapi :: SetCursor(int Lin)
{
        Lin--;
        WORD Loc = LOWORD(Lin*80+2);                    // fixed coloumn = 2
        Wait();
        ret_code = HLL_SetCursor(hWnd,Loc);
        if(ret_code)    return(onError(HLL_SETCURSOR,ret_code));
        return HLL_SUCCESS;
}

CHllapi :: Disconnect()
{
```

```
        ret_code = HLL_DisconnectPS(hWnd);
        if(ret_code)    return(onError(HLL_DISCONNECTPS,ret_code));
        return HLL_SUCCESS;
}

CHllapi :: onError(int funccode, int ret_code)
{
        AnsiString abuff;

        abuff = "Function : " + (AnsiString) funccode + "   Return Kodu : "
                + (AnsiString) ret_code;
        txt->Text = abuff;

        switch(ret_code)
        {
                case HLL_INVALIDPSID :          //1
                        Mainf->Memo1->Lines->Add("BAGLANTI KOPTU.");
                        Mainf->Memo1->Lines->Add("CIKIS'A BASINIZ!!");
                        break;

                case HLL_INVALIDPARAMETER    :      // 2
                        break;

                case HLL_TIMEOUT             :      // 4
                        Mainf->Memo1->Lines->Add(
                                "Sistem cevap verme suresini asti.");
                        break;

                case HLL_PSLOCKED            :      // 5
                        Mainf->Memo1->Lines->Add("Sistem kilitlendi.");
                        break;

                case HLL_FIELDSIZEMISMATCH   :      // 6
                        break;

                case HLL_INVALIDPSPOSITION   :      // 7
                        break;

                case HLL_NOPRIORSTARTKEYSTROKE :    // 8
                        break;

                case HLL_SYSTEMERROR         :      // 9
                        Mainf->Memo1->Lines->Add("Sistem hatasi.");
                        break;
```

```
                case HLL_RESOURCEUNAVAILABLE  :      // 11
                        break;

                case 19                                 :      // 19
                        break;
                case HLL_NOSUCHFIELD          :      // 24
                        break;
        }
        return(1);
}
```

UTILITY.H

```
//---------------------------------------------------------------------------
#ifndef UtilityH
#define UtilityH

void timer(clock_t delay);
BOOL showMesajtlx();
BOOL showHatali();
char* crBul(char* pSatir);
void finish();

//---------------------------------------------------------------------------
#endif
```

UTILITY.CPP

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Main.h"

//---------------------------------------------------------------------------
#pragma package(smart_init)

void timer(clock_t delay)
{
        clock_t goal;
        goal = delay + clock();
        while(goal > clock())
          ;
}
```

```
BOOL showMesajtlx()
{
        FILE *mesajtlx;
        char record[RECLEN];

        if ((mesajtlx = fopen(MESAJTLX,"r")) == NULL){
                Application->MessageBox("Mesajlar.tlx acilamiyor!!!",
                        "Dosya Hatasi", MB_OK);
                return FALSE;
        }
        Mainf->Memo1->Lines->Add("          ");
        Mainf->Memo1->Lines->Add("Mesajlar.tlx");
        Mainf->Memo1->Lines->Add("          ");

        while(fgets(record, RECLEN, mesajtlx)){
                crBul(record);
                Mainf->Memo1->Lines->Add(record);
        }
        fclose(mesajtlx);
        Mainf->Memo1->Lines->Add("          ");
        return TRUE;
}

BOOL showHatali()
{
        FILE *hatali;
        char record[HATALIREC];
        int mm;

        if ((hatali = fopen(HATALI,"r")) == NULL){
                Application->MessageBox("Hatali.lst acilamiyor!!!",
                        "Dosya Hatasi", MB_OK);
                return FALSE;
        }

        Mainf->Memo1->Lines->Add("          ");
        Mainf->Memo1->Lines->Add("Hatali.lst");
        Mainf->Memo1->Lines->Add("          ");

        while(fgets(record, HATALIREC, hatali)){
                crBul(record);
                AnsiString Rec(record);
                mm = Rec.Pos(".");
                Rec = Rec.SubString(1, mm+3);
                Mainf->Memo1->Lines->Add(Rec.c_str());
```

```
        }
        fclose(hatali);
        Mainf->Memo1->Lines->Add("          ");
        return TRUE;
}

char* crBul(char* pSatir)
{
        char *p;
        for(p=pSatir; *p; p++)
        if(*p == CR || *p == LF)                *p = ' ';
        return pSatir;
}

void finish()
{
        if(!resetHllapi())              exit(EXIT_FAILURE);
        exit(EXIT_SUCCESS);
}
```

ÖZGEÇMİŞ

Yazar, 30.08.1974 tarihinde İstanbul'da doğmuştur. Orta öğrenimini Özel Çavuşoğlu Lisesi'nde tamamlayarak 1991 yılında İstanbul Teknik Üniversitesi Fen-Edebiyat Fakültesi Matematik Mühendisliği Bölümü'nde lisans eğitimine başlamıştır. 1996 yılında Matematik Mühendisi ünvanı alarak mezun olmuştur. Aynı yıl İstanbul Teknik Üniversitesi Mühendislik Bilimleri Anabilim Dalı Sistem Analizi Programı'nda yüksek öğrenimine başlamıştır. Halen THY Elektronik Bilgi İşlem Merkezi'nde analist programcılık görevini sürdürmektedir.