T.C.
MİLLÎ EĞİTİM BAKANLIĞI

MEGEP

(MESLEKİ EĞİTİM VE ÖĞRETİM SİSTEMİNİN
GÜÇLENDİRİLMESİ PROJESİ)

BİLİŞİM TEKNOLOJİLERİ

PROGRAMLAMA TEMELLERİ

ANKARA 2007

Milli Eğitim Bakanlığı tarafından geliştirilen modüller;

- Talim ve Terbiye Kurulu Başkanlığının 02.06.2006 tarih ve 269 sayılı Kararı ile onaylanan, Mesleki ve Teknik Eğitim Okul ve Kurumlarında kademeli olarak yaygınlaştırılan 42 alan ve 192 dala ait çerçeve öğretim programlarında amaçlanan mesleki yeterlikleri kazandırmaya yönelik geliştirilmiş öğretim materyalleridir (Ders Notlarıdır).

- Modüller, bireylere mesleki yeterlik kazandırmak ve bireysel öğrenmeye rehberlik etmek amacıyla öğrenme materyali olarak hazırlanmış, denenmek ve geliştirilmek üzere Mesleki ve Teknik Eğitim Okul ve Kurumlarında uygulanmaya başlanmıştır.

- Modüller teknolojik gelişmelere paralel olarak, amaçlanan yeterliği kazandırmak koşulu ile eğitim öğretim sırasında geliştirilebilir ve yapılması önerilen değişiklikler Bakanlıkta ilgili birime bildirilir.

- Örgün ve yaygın eğitim kurumları, işletmeler ve kendi kendine mesleki yeterlik kazanmak isteyen bireyler modüllere internet üzerinden ulaşılabilirler.

- Basılmış modüller, eğitim kurumlarında öğrencilere ücretsiz olarak dağıtılır.

- Modüller hiçbir şekilde ticari amaçla kullanılamaz ve ücret karşılığında satılamaz.

İÇİNDEKİLER

İÇİNDEKİLER i
AÇIKLAMALAR ii
GİRİŞ 1
ÖĞRENME FAALİYETİ - 1 3
1. PROGRAMLAMA DİLLERİ 3
1.1. Bilgisayar Programı İçin Neler Bilmeliyim? 6
1.2. Programlama Dili 6
1.3. Neden Birçok Programlama Dili Vardır? 6
1.4. Hızlı Uygulama Geliştirme Ortamları 11
1.5. Veri Tabanı (Database) Programcılığı 12
1.6. Betik (Script) Programcılığı 12
1.7. Web Programcılığı 13
UYGULAMA FAALİYETİ 15
ÖLÇME VE DEĞERLENDİRME 16
ÖĞRENME FAALİYETİ - 2 18
2. PROGRAM YAZIMI 18
2.1. Prototip (Kalıp) Oluşturma 20
2.2. Sahte Kod ve Algoritma Yazma 20
2.3. Akış Şemaları 23
2.4. Karar Tabloları 27
2.5. Bir Programın Hayat Döngüsü 29
UYGULAMA FAALİYETİ 32
ÖLÇME VE DEĞERLENDİRME 33
ÖĞRENME FAALİYETİ - 3 35
3. PROGRAMLAMA ARAÇLARI 35
3.1. Düzenleyici 36
3.2. Derleyici 36
3.3. Böcek Ayıklayıcı (Debugger) 37
3.4. Yardım ve Kurulum 38
UYGULAMA FAALİYETİ 44
ÖLÇME VE DEĞERLENDİRME 45
MODÜL DEĞERLENDİRME 48
CEVAP ANAHTARLARI 51
SÖZLÜK 54
ÖNERİLEN KAYNAKLAR 56
KAYNAKÇA 57

AÇIKLAMALAR

KOD	**481BB0024**
ALAN	**Bilişim Teknolojileri**
DAL/MESLEK	**Alan Ortak**
MODÜLÜN ADI	**Programlama Temelleri**
MODÜLÜN TANIMI	Programlama altyapısını ve temellerini oluşturan öğrenme materyalidir.
SÜRE	40/16
ÖN KOŞUL	
YETERLİK	Bilgisayarda program yazmaya hazırlık yapmak
MODÜLÜN AMACI	**Genel Amaç** Gerekli ortam sağlandığında, programlama dillerini tanıyıp, bilgisayarda bir problemin çözüm aşamalarını yaparak, program yazmaya hazırlık yapabileceksiniz. **Amaçlar** 1. Programlama dillerini işlevlerine göre ayırt edebileceksiniz. 2. Program yazmaya hazırlık yapabileceksiniz. 3. Program yazım araçlarını kullanabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	Bilgisayar laboratuvarı ve bu ortamda bulunan; bilgisayar, yazıcı, bilgisayar masaları, kâğıt, kalem, lisanslı işletim sistemi programı ve akış diyagramı sembolleri ile ilgili panolar.
ÖLÇME VE DEĞERLENDİRME	Her faaliyet sonrasında o faaliyetle ilgili değerlendirme soruları ile kendi kendinizi değerlendireceksiniz. Modül içinde ve sonunda verilen öğretici sorularla edindiğiniz bilgileri pekiştirecek, uygulama örneklerini ve testleri gerekli süre içinde tamamlayarak etkili öğrenmeyi gerçekleştireceksiniz. Sırasıyla araştırma yaparak, grup çalışmalarına katılarak ve en son aşamada alan öğretmenlerine danışarak ölçme ve değerlendirme uygulamalarını gerçekleştireceksiniz..

GİRİŞ

Sevgili Öğrenci,

Her şeyden önce herkes bir programlama dilini öğrenebilir. Bilgisayar programlama yüksek bir zekâ ve matematik bilgisi gerektirmez. Sadece asla vazgeçmeme sabrı ve öğrenme isteği yeterlidir.

Programlama bir hünerdir. Bazı insanlar doğal olarak diğerlerinden daha iyidir, ama herkes pratik yaparak iyi olabilir. Başaramamaktan korkmak yerine, kendinizi bu maharete vererek, öğrenmek için uğraşın. Programlama eğlencelidir, fakat yanlış çalışma yöntemleriyle sinir bozucu da olabilir ve zamanınızın boşa geçmesine neden olabilir. Bu sebeple bu modülleri takip ederek, en az sıkıntı ve en yüksek memnuniyet ile programlamayı öğreneceksiniz.

Programlamada, bir problemin çözüm aşamalarının sıralı bir şekilde yazılmasına "Algoritma", bu aşamaların şekillerle gösterilmesine ise "Akış diyagramı" denir. Algoritmayı, günlük hayattaki bir probleminizi çözerken yapmanız gereken uygulamaları belli bir düzene sokma işlemine benzetebilirsiniz.

Algoritma, programlamanın temelidir. Önce çözüm belirlenir sonra kullanılacak programlama diline uygun komutlarla program yazılımı tamamlanır. Algoritma bir binanın temeli gibidir. Temeli sağlamsa bina ayakta durur. Bu yüzden algoritmanın iyi tasarlanması, programın temelinin iyi oluşturulması gerekmektedir.

Bu modül sonunda; istediğiniz herhangi bir problemin bilgisayardaki çözüm aşamalarını yazabileceksiniz. Çözüm aşamalarını belirledikten sonra istediğiniz bir programlama dilini kullanarak gerekli yazılımı oluşturulabileceksiniz. Akış diyagramları ile yazdığınız algoritmayı şekillerle daha kolay, anlaşılır ve takibi kolay kılacaksınız.

ÖĞRENME FAALİYETİ–1

AMAÇ

Programlama dillerini işlevlerine göre ayırt edebileceksiniz

ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Bilgisayarınızdaki kurulu programların genel olarak kategorilerini belirleyip, bir liste hazırlayınız. Mesela müzik programları, resim programları, yardımcı programlar gibi…
- Kullandığınız programların hangi programlama dilleri ile yapıldığını araştırınız.
- Size göre çok ilginç olan bir programın özelliklerini anlatınız. Ne işe yaradığını ve nereden edinilebileceğini not ediniz.
- Farklı programlama dillerinde "Merhaba Dünya – Hello World" program örnekleri bulunuz. Dilin adını ve örnek programı bir kâğıda yazınız.

Konularda bazı kelimelerin altı mavi dalgalı olarak çizilidir. Bu konuları araştırmanız tavsiye edilir.

Not: Her dilin kendine göre avantajı bulunmaktadır. Modülde belli yerlerde verilen örnekleri yaptığınızda o dilleri öğrenmiş olmayacaksınız. Asıl amaç, anlatılmak istenen konunun uygulanmasıdır, somut ve anlaşılır hale gelmesidir.

1. PROGRAMLAMA DİLLERİ

Bilgisayar çok karmaşık bir elektronik cihazdır, şimdilik bizi ilgilendiren kısım onun çalışma prensibi değil, programlama kısmıdır.

Bilgisayar öğrendiğini unutmaz, eğer iyi programlarsanız kusursuz olarak işlemleri yapar, yorulmadan hep aynı işlemi tekrar yapabilir. Programlar bilgisayarın tüm kaynaklarına erişebilir, tabii onu yazan programcı izin vermiş ise. Bilgisayar, bir konuda yorum yapamaz, yeni durumlara uyum sağlamak için çaba sarf etmez.

İnsan unutkandır, hata yapabilir, yorulur ve beyninin tamamını kullanamaz, moral durumu değişebilir ve duygusal olarak etkilenir. İnsanların en büyük avantajı, yeni durumlar karşısında bocalasa bile zamanla uyum sağlayabilmesidir. Başına gelen olaylardan ders çıkartabilir. Tek başına birçok sorunu çözebilirler. Bilgisayar ise her zaman aynı tepkiyi verir, kendini geliştiremez. İnsan tarafından kontrol edilmedikçe etkinliklerini değiştirmezler.

Bilgisayar; ekonomi, bilim, mühendislik, eğitim ve askeri alanlarda yardımcı olması için üretilmiştir. Çok karmaşık formüllerin sonucunu kısa zamanda elde etmek için programlar yazılmıştır. Özellikle İkinci Dünya Savaşı veri şifreleme ve silahların hedefi daha doğru bulması gibi konular sebebi ile bilgisayarın gelişimi hızlanmıştır.

Bir programlama dilini neden öğreniriz? Bu sorunun cevabı, eğlence için, bir ihtiyacı gidermek için, kariyer için veya zekânızı kanıtlamak için olabilir. Para kazanmak her ne kadar birincil hedef gibi görünse de, eğer işinizi severek yapmıyor iseniz, ne kadar kazandığınızın pek önemi olmaz.

Eğer bilgisayara ne yapması gerektiğini söylemezseniz, hiçbir şey icra etmez. Bilgisayara yaptırmak istediğiniz şeyi iki şekilde gerçekleştirebilirsiniz:

- Adım adım bir program yazarak
- Uygun bir program satın alarak

Bazı programların giriş ve çıkışları:

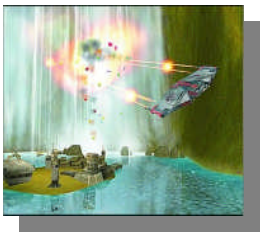
Tür	Giriş	İşlem	Çıkış
Kelime İşlemci	Klavyeden girilen karakterler	Yazıyı biçimlendirir, yazımını kontrol eder	Düzgünce ekrandan veya yazıcıdan çıkış
Oyun	Fare, klavye ve oyun çubuğu tuş basımları	Ekranda hızlı bir şekilde bir animasyon hesaplama	Ekrandaki figürler
Muhasebe Programı	Şu anki ve geçmiş fiyatlar	Piyasadaki fiyat etkilerini tanımlama	Gelecekteki ürün fiyatı
Web Tarayıcı	HTML kodları	Kodları resim ve yazıya dönüştürme	Web sayfasını ekranda gösterme

İyi bir programın temel özellikleri şunlardır:

Doğruluk : Verilen görevlerin tam olarak yerine getirilmesidir.
Dayanıklılık : Beklenmedik hatalardan dolayı programın çalışması kesilmemelidir.
Genişletilebilme : İleri aşamalarda görevlerin değişikliği veya yenilerinin eklenmesi kolay olmalıdır.
Basitlik : Karmaşık tasarımlardan kaçınmak gerekir.
Modülerlik : Program kodları başka programlar içinde de kullanılabilmelidir.
Uyumluluk : Başka bilgisayar ve sistemlerde çalışabilmelidir.
Kontrol edilebilirlik : Hata olabilecek yerlere açıklayıcı hata mesajları konulmalıdır.
Kolay kullanım : Kullanıcı ara birimi kolay olmalı ve rahat öğrenilebilmelidir.
Parçalanabilirlik : Problemin küçük parçalara ayrılarak yazılmasıdır.
Anlaşılırlık : Başkasının yazdığı program elden geçirilirken rahatça okunabilmelidir.
Koruma : Modüller birbirlerine müdahale etmemelidirler.

Temel olarak bir algoritma, bilgisayara belli bir problemin çözümünü anlatmaktır. Dünya problemlerden oluştuğuna göre, pratikte insanların yazabileceği program sayısı ve çeşitliliğinin bitmesi çok zordur. Büyük bir problemin bilgisayara yaptırılması için, genellikle küçük parçalara bölmek gereklidir.

Örneğin bir oyunda şu adımları çözmek gerekebilir:



- Kullanıcının bir nesneyi (araba, adam, uzay gemisi…) nasıl hareket ettirebileceği
- Nesnenin duvar, uçurum veya çevredeki başka nesnelere göre çarpmadan, düşmeden nasıl hareket edeceği
- Gerçekçi olarak çevre tasarımı yapmak
- Gelen merminin nereye çarptığını bulmak ve oyuncunun sağlık durumunu ekrana yazmak

Programlama genellikle zor değildir, ama zaman harcayan bir iştir. Bir pinpon oyununu yazmak, bir savaş uçağının simülasyonunu yapmaktan daha kolaydır. Eğer adım adım bir yeri tarif edebiliyorsanız, program yazabilirsiniz. Bir bilgisayar aptal olduğuna göre ona ne yapacağını adım adım anlatmanız gereklidir.

Arkadaşınıza evinizin yolunu tarif ederken:

- Yayla mahallesine git
- Çınar sokağına git
- Trafik ışıklarından sağa dön
- İki sokak ileride sola dön

Bir insan için kolay olan bu adımlar bilgisayarda yazıldığında hiçbir anlam ifade etmez:

- Yayla mahallesine nasıl gideceğim?
- Çınar sokağını nasıl belirleyeceğim?
- Işıklardan ne kadar sonra sağa döneceğim?
- Sola dönünce arabayı nasıl park edeceğim?

Programcılık

Programlar her zaman işe yaramayabilir. 1980'li yıllarda Amerikan ordusu "Teğmen York" isminde bir pilotsuz uçan uçak savar planlamıştı. Basit olarak amaç, düşman uçağını bulup yok etmekti. Milyonlarca dolar harcandıktan, saatlerce program yazıldıktan ve test edildikten sonra programcılar çalışacağına emin oldular. En üst rütbeli subaylar ve görevliler önünde, deneme için hazırlık yapıldı. Maalesef deneme sırasında, "Teğmen York" tüm silahlarını düşmana değil, komutanların üzerine doğrultmuştu. Neyse ki insanlar etrafa dağılırken, füzelerini ateşlememişti! Bu korkunç denemeden sonra proje rafa kaldırıldı.

1.1. Bilgisayar Programı İçin Neler Bilmeliyim?

Bir programı kullanmaktan çok, yazma konusunda **istekli** iseniz, zaten program yazmak için gerekli şeye sahipsiniz demektir.

- **İstek**: Önünüze ne kadar engel çıksa da, isteğiniz varsa öğrenirsiniz. (Kanuni olmayan bir şey ile ilgileniyorsanız, hapishanede geçirilecek zamanınız olabilir!)
- **Meraklılık**: Bu sayede öğreneceğiniz dil size angarya gibi gelmez.
- **Hayal gücü**: Böylece daha ilginç ve faydalı program yapabilirsiniz.

1.2. Programlama Dili

Bilgisayar Türkçe, İngilizce veya başka bir dilden anlamaz. Bilgisayarın fonksiyonel bir beyni olmadığı için insanlar komutlar yazmalıdır. Bu özel dile "**programlama dili**" denir.

Komutlar bir araya gelerek "program"ı meydana getirir. Belli bir dil ile yazılmış komutlara "**kaynak kod**" da diyebiliriz.

1.3. Neden Birçok Programlama Dili Vardır?

Her programlama dili özel bir amaca hizmet eder. İnsanlar farklı problemleri çözmek için değişik programlama dilleri yazmışlardır. Temelde bilgisayarlar 1 ve 0'lardan oluşan ikilik sayı* sistemindeki dili anlarlar. Buna "makine dili" denir:

```
0010 1010 0001 1101
0011 1100 1010 1111
0101 0110 1101 0101
1101 1111 0010 1001
```

Makine dilinin dezavantajları, kodları yazarken hata yapılma olasılığının fazla olması ve yazımının uzun sürmesidir. Makine dilinin daha rahat programlanması için 1950'li yıllarda "assembly dili" geliştirilmiştir. "Assembly dili" basit, hatırlanması kolay deyimlerden oluşur.

Programcılar 1 ve 0 ile program yazma yerine "assembly dili"ni geliştirmişlerdir, bilgisayarın bu yazılanlarla ilgili hiçbir fikri yoktur. Bu sebeple programcılar "assembly dil" komutlarını "makine dili"ne çeviren programlar yazmışlardır. Bu programlara **"makine dili çeviricisi - assembler"** denir. Böylece "assembly dili" ile yazılmış bir kod, bilgisayarın anlayabileceği "makine dili"ne dönüşmüş olur.

Not: Bir programı ilk denemede çalıştırmak zordur. Genel davranış olarak programı okunaklı yazarsanız, hatalara fırsat vermez, ileride bir eklenti yapmak istediğinizde zorlanmazsınız.

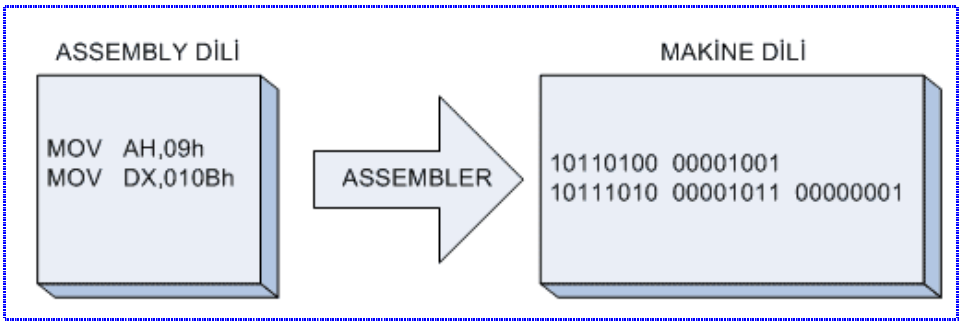
---

* Bir çevrim örneği:

$$(1100)_2 = 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 0 * 2^0 = 8 + 4 = (12)_{10}$$

Assembly dilinde kod örneği

```
Title Yazi Programi
dosseg
.model small
.stack 100h
.data
my_message db 'Selam!',0dh,0ah, '$'
.code
main proc
          mov ax,@data
          mov ds,ax
          mov ah,9
          mov dx,offset my_message
          int 21h
          mov ax,4C00h
          int 21h
main endp
end main
```



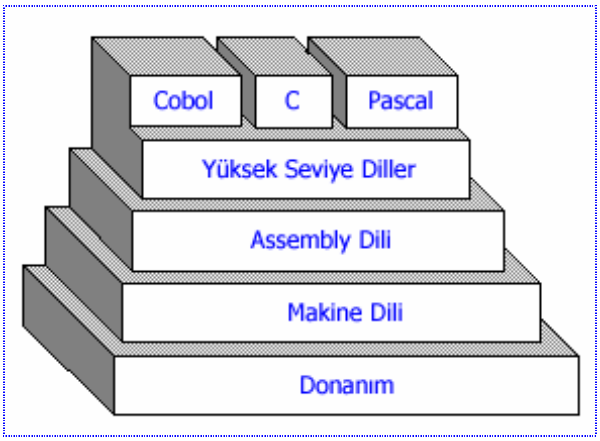
Resim 1.1: Bilgisayarın anladığı dil; makine dili

Not: Intel uyumlu ve Microsoft'un MS DOS ve Windows işletim sistemlerine uygun bir "assembly dili" ile burada örnek verdik. Kişisel bilgisayarlar (PC), PowerPC (Macintosh), PIC ve 8051 gibi işlemci ve mikro denetleyicilerin kendilerine has dilleri ve çevirici programları vardır.

Temel olarak makine dilinde; mantıksal işlemler, aritmetik işlemler, dallanma işlemleri ve veri hareket işlemleri yapabiliriz. Bu işlemleri kullanarak ister basit ister çok karmaşık programlar yazabilirsiniz.

"Assembly dili" makine dilinden daha rahat yazılmasına rağmen, hala bazı dezavantajlara sahiptir:

- Yavaş ve çok yer kaplayan programlar oluşur. (bu satır hatalı assembly yazılan programlar az yer kaplar ve yavaş çalışmazlar)
- Başka dile çevrilmeleri zordur.
- Çok uzun program yazımına elverişli değildir.



Resim 1.2: Dillerin genel görünümleri

Programcılar donanıma erişimi daha kolay olan, okunaklı, yazımı ve düzenlemesi kolay olan bir dile ihtiyaç duymuşlardır. Bunun üzerine **C** (1972 doğum yılı) taşınabilir[†] dili geliştirilmiştir. **Cobol** (1959) ve **Fortran** (1957) gibi birçok diller de vardır ama hala günümüzde yaygın olarak kullanılan C dili olmuştur.

C dilinde kod örneği

```
main()
{
printf ("Selam!\n");
}
```

Ekrana "Selam!" yazan bu örnek İngilizce diline benzer kodlamaya sahiptir. İnsanların konuşma diline yakın olan dillere "**yüksek seviye dil**" denilmiştir.

Aslında programlama dilinin İngilizce olması şart değildir. Kendi dilimizde olan örnek program **Tupol**[‡] ile yazılmıştır.

Tupol dilinde kod örneği

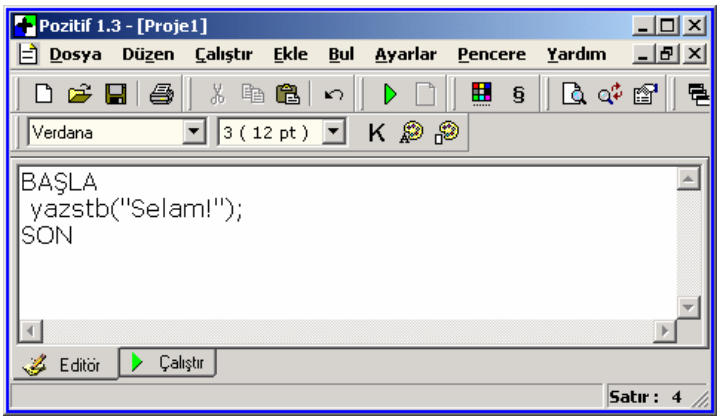
```
PROGRAM SelamProgrami;
Basla
          Yazi(#i,"Selam!");
Bitti.
```

Resim 1.3'te başka bir Türkçe programlama dili olan Pozitif[§] dilinden örnek var.

[†] Taşınabilir: Tasarlandığı bilgisayarın haricinde, yani donanımdan bağımsız olarak diğer bilgisayar türlerinde de çalışabilme özelliğidir.

[‡] "1995 yılında iki genç bilgisayar mühendisi adayı (Savaş KÖSE ve Mehmet AKIN) tarafından lisans derslerinin arasında 4–5 ay gibi kısa bir sürede yazılan TUPOL 2.0 ilk, belki de tek Türkçe derleyici olması, azim ve çalışmanın sonuçlarını göstermesi açısından önemli bir örnektir."

[§] "Pozitif bazı matematiksel işlemleri, metinler üzerinde oynamayı, kullanıcıdan bilgi girişi alarak bunlar üzerinde işlem yapabilmeyi sağlayan bir dildir." Birkan KUYUMCU



Resim 1.3: Pozitif Türkçe programlama dili ile örnek program yazımı

Her ne kadar kendi dilimizde program yapmak gurur ve heyecan verici olsa da, ileride göreceğiniz bir dilin temel gereklilikleri sebebi ile bu dillerde büyük (karmaşık) programlar yapmak zordur. Programlamaya ısınmak için, ilk denemelerinizi bu programlama dillerinde yapabilirsiniz.

Bir dilin okunaklı ve kolay yazılabilir olması dışında, bilgisayarın donanımına ulaşması ve başka bilgisayarda da rahatlıkla çalışması gereklidir. Yüksek seviye bir dili "makine diline" çeviren programlara **"derleyici – compiler"** denir. Derleyiciyi aynen Fransızca bir kitabın Türkçeye çevrilmesini sağlamak gibi düşünebiliriz. Tabii insan dilinin başka dile çevrimi çok daha zor bir işlemdir.

Eğer uygun bir derleyiciniz var ise programınızın kaynak kodunu başka bilgisayarda derleyerek çalıştırabilirsiniz. Örneğin, Macintosh'ta yazılan bir programın kodlarını Windows'ta biraz değiştirerek kullanabilirsiniz.

C programlama dili diğer Cobol ve Fortran gibi dillerden daha basit olduğu için, C dilini makine diline çeviren birçok derleyici yazılmıştır. C dili böylece öyle çok yaygınlaşmıştır ki, bu dile uyumlu olan diller ortaya çıkmıştır. Mesela C++ (*c plus plus*), Java, Perl, Python ve C# (*c sharp*) gibi. Şu anda kullanılan tanınmış, tanınmamış birçok program C veya C++ ile yapılmıştır: Windows, Unix, Microsoft Office gibi...

Not: C'nin çok güçlü olması bazı kötülükleri de beraberinde getirir: Tüm sistem kaynaklarına erişmek gibi. Bu kullanmasını bilmeyen birinin eline elektrikli testere vermek veya el bombasını vermek gibidir… Yanlış yazılan kod, bilgisayarın göçmesine sebep olabilir. Özellikle eski DOS günlerinde programdaki hatalardan dolayı bilgisayarın kilitlenmesi çok sık olan bir durumdu. Yeni işletim sistemleri, program hatalarında bilgisayarın kilitlenmemesi için önlemler almışlardır.

Programlama dilleri özel amaçlar için yazıldıkları için o dilde her istediğinizi yapmak zordur. Mesela Fortran matematik hesaplamalar için yapılmıştır, bu dilde bir işletim sistemi yazmak gereksiz yere çok uzun zaman alan bir işlem olur.

Mühendisler diğer insanların rahatça program yazmaları için **Basic** (*Beginners All Purpose Symbolic Instruction Code* - 1964) ve **Pascal** (1971) gibi diller üretmişlerdir. Basit bir dilden yola çıkılarak istenen dile geçiş yapılabilir.

Basic dilinde kod örneği

```
PRINT "Selam!"
```

Gördüğünüz gibi ekrana bir mesaj yazmak için Basic'te sadece bir satır kod yeterli oluyor. Bu dil sayesinde insanlar kodun yazımı ile uğraşmaya değil, amaçladıkları işe odaklanabiliyorlar.

C ve Basic arasında kalan Pascal ise biraz daha iyi görünümlü, yapısal program yazmayı sağlıyor:

Pascal dilinde kod örneği

```
Program Message;
Begin
          Writeln ('Selam!');
End.
```

Lisp 1950'li yıllarda yapılmış bir derlenebilen mantık programlama dilidir. **Prolog** ("*programming in logic*" kelimelerinin kısaltması) adındaki 1972 yılında ortaya çıkan dil ise, "mantık programlama dili" olarak tarif edilebilir. Komut dizileri yerine mantık deyimleri ile program yazılır. Genellikle yapay zekâ ile ilgili konularda kullanılır. Yazımı ve kuralları çok basittir.

Yüksek seviyeli dillerin genel özelliklerini şöyle özetleyebiliriz:

- "Makine diline" göre daha şişkin ve yavaş kod meydana getirirler.
- Tüm sistem kaynaklarına ulaşılamayabilir, tabii amacınız disk yardımcı programı (Norton SystemWorks…) gibi program yazmak değilse…
- Bir derleyici gereklidir.
- Kısa zamanda program yazmaya başlanır.
- Öğrenme ve ustalaşma için geçen zaman fazla değildir.
- Yanlışlıkla sistem kaynaklarının bozulmaması için kalkanları vardır.
- Okuması ve değiştirmesi kolaydır.
- Başka bilgisayar çeşitlerinde de çalışabilirler, yani taşınabilirdirler.

Sık sorulan sorular:

1. Assembly dili ile yüksek seviye dil arasındaki hız farkı ne kadardır?
Cevap: Assembly dili yaklaşık 2 ile 20 kat arasında daha hızlıdır.

2. Assembly dili ile yüksek seviye dil arasındaki program tasarım süresi ne kadardır?
Cevap: Assembly dili yaklaşık 10 ile 100 kat arasında daha yavaş sürede tamamlanır.

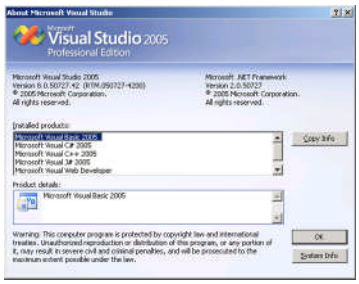
3. Derleyiciler hangi dilde geliştirilmişlerdir?
Cevap:Önceleri assembly dili ile yazılmışlardır. Daha sonra yüksek seviye diller güçlendikçe bu diller kullanılarak derleyiciler yazılmıştır.

1.4. Hızlı Uygulama Geliştirme Ortamları

Eski zamanlarda fare, grafik ekran imkânı olmadığı için “metin tabanlı” ortamlarda çalışılırdı. Pencere ve düğme gibi kavramlar ise, 1984 yılında Apple "System 1" ve 1992 yıllarında Microsoft Windows 3.1’in ortaya çıkması ile günlük hayatımıza girdi.

Pencereler, kaydırma çubukları, araç çubukları, menüler programcının daha hızlı bir şekilde program yapmasına sebep olduğu için, bu ortamlara “Hızlı Uygulama Geliştirme – *Rapid Application Development* (RAD)” denilmiştir. **Görsel Programlama Dilleri** (*Visual Programming Languages*) olarak da adlandırılırlar. Programcı oluşturacağı programın ara birimini, istediği gibi tasarlayabilir ve bu ara birimi işlevsel hâle getiren kod kısmını yazabilir.

Popüler RAD dilleri: Visual Basic, Delphi, JBuilder ve Visual C#.

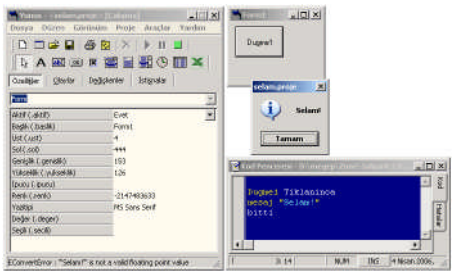


Resim 1.4: Visual Studio 2005 programının hakkında kısmı

RAD ile programcı, temel programlama dilleri olan C, Basic ve Pascal gibi dilleri kullanarak, hızlı bir şekilde uygulama geliştirebilir. Kullanıcı ara birimi geliştirmek kolaydır. C ve Basic bilen için görsel programlama diline geçiş çok kolaydır.

Maalesef RAD ile dilin taşınabilirliği azalır. Yani Windows ortamında Visual Basic ile yazılan programın, Linux’ta çalışma ihtimali azdır. Tahmin ettiğiniz gibi, artık görsel programlama ile yaptığınız programlar daha çok disk alanı kaplayacak ve daha yavaş çalışacaklar.

Türkçe görsel programlama diline örnek Yunus’tur**.



Resim 1.5: Yunus ekran görüntüsü

** “Çoğunlukla Visual Basic benzeri kodlama sistemimiz var. Aslında hoşuma giden Pascal ve C++ komutları ekledim. Sonuçta Türkçe veya başka bir dile özelleştirilebilen basit bir eğitim amaçlı script (betik) dilimiz oldu.” Tarık BAĞRIYANIK

1.5. Veri Tabanı (Database) Programcılığı

C ve benzeri diller genel amaçlı diller olduğu için, bu diller ile oyun, muhasebe programı veya bir kelime işlemci programı yapılabilir. Bilgisayarların en çok kullanım alanı ise, bilgi kaydetme ve düzeltmedir. İsimler, adresler, elektronik posta adresleri, telefon numaraları, iş deneyimleri, muhasebe geçmişleri gibi bilgiler veri tabanlarında saklanır. Neredeyse tüm firmalar veri tabanı sayesinde işlerini daha rahat yürütürler.

Kimse garip komutlar ile kayıt, okuma ve silme işlemlerini yapmak istemez. Veritabanını kolaylaştırmak için "veritabanı (*database*) programcılığı" geliştirilmiştir. Veri tabanı programlama ile değişik firmaların isteklerine göre özel veri tabanları oluşturabilirsiniz.

En çok kullanılan veri tabanı programları Microsoft Access, MSSQL ve MySQL'dir. Verileri düzenleme için kendilerine has dilleri vardır. Bu dile SQL (*Structured Query Language* – Yapısal Sorgu Dili) denir. dbase, FileMaker ve FoxPro programları ise artık pek kullanılmıyor. Yine SQL tabanlı Oracle ise, çok büyük veri tabanları barındıran, güvenlik ve hız gerektiren uygulamalarda kullanılır.

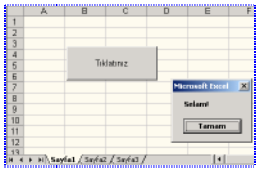
Veri tabanı programcılığının dezavantajları şunlardır:

- Veri tabanı programları, veri tabanını işleyen program sistemde kurulu değilse çalıştırılamazlar.
- Sadece veri tabanı programını kullanarak antivirüs, oyun, kelime işlemci gibi program yapamazsınız.

1.6. Betik (Script) Programcılığı

Birçok program kendi programlama dilinin kullanılması imkânını sağlar. Örneğin Word ve Excel içinde, neredeyse Visual Basic'e benzeyen, Visual Basic for Applications (VBA) adı verilen bir dil vardır.

Office programları içindeki makrolar (VBA) kullanıcıya yardımcı işlevler sağlar:



Resim 1.6: Excel'de makro örneği

VBA dilinde kod örneği

```
Private Sub CommandButton1_Click()
 MsgBox "Selam!"
End Sub
```

Macintosh işletim sisteminde bulunan AppleScript ile basit bir örnek görelim:

AppleScript dilinde kod örneği

```
on DisplayMessage()
          display dialog "Selam!" buttons {"Tamam"}
end DisplayMessage
DisplayMessage()
```

Windows işletim sisteminde bulunan yığın dosyası (*.bat) ile basit bir örnek görelim:

Batch dilinde kod örneği

```
@Echo off
Echo Selam!
Pause
```

Betik dilleri normalde diğer programlama dillerinden daha kolaydır. Bu diller ile kullanıcıya yardımcı bazı işlemler yapabiliriz:

- Var olan bir programı özelleştirebiliriz.
- Çok az program yazarak, karmaşık bir programı kullanışlı hale getirebiliriz.

Şunları unutmayınız:

- Betik dili ana programa bağımlıdır, sadece ana programın kurulu olduğu bilgisayarlarda yazdığınız program çalışabilir.
- Programınızın satış ve dağıtımı zordur. Almak isteyen ana programı da satın almak zorunda kalır.
- Genel programlama dillerinden daha kısıtlı imkânlarınız vardır.

1.7. Web Programcılığı

Bazı anlarda okunaklılığı azaltsa da, günümüz internet siteleri bol grafik, efekt ve animasyonlarla süslüdür. Kolay web programcılığı için **HTML** (*HyperText Markup Language* – Hiper Yazı İşaret Dili) icat edilmiştir.

Bir sitenin HTML olarak kaynağını incelediğinizde çok karmaşık ve dağınık gelebilir. HTML ile site tasarımı çok zor olduğu için JavaScript gibi web sayfası yardımcı programlama dilleri yapılmıştır. Etkileşimli site yapmak, kullanıcının formlara girdiği verileri kontrol etmek, site içine oyun eklemek isterseniz, bunları JavaScript, Java, CGI veya Macromedia Flash ile yapabilirsiniz.

JavaScript dilinde kod örneği

```
<script language="javascript">
 alert("Selam!");
</script>
```

PHP dilinde kod örneği

```
<?
  echo "Selam!";
?>
```

ASP dilinde kod örneği

```
<%
  Response.Write "Selam!"
%>
```

Web programlama dilleri, sitenin ziyaretçi sayısını artırır ve ziyaretçinin uzun süre sitede kalmalarını sağlar. Öğrenmesi kolaydır ve dünyanın herhangi bir yerinden siteye (uygulamaya) ulaşıp, değişiklikler yapabilme imkânı vardır.

Dezavantajları:

- Tüm web tarayıcıları JavaScript ve Java desteklemez. (Daha eski tarayıcılar gibi)
- Bağlantı hızını düşürür.
- İnternet bağlantısı yok ise sayfanın görüntülenmesi mümkün olmayabilir.

Sonuç olarak her konuda bir işe yarayan "mükemmel" dil yoktur. Profesyonel olarak program yazmak istiyorsanız, C# gibi yüksek seviye bir dil ve SQL gibi veri tabanı dili öğrenmelisiniz. İş bulmakta zorlanmazsınız. Eğer sizden site yapılması isteniyor ise Java, JavaScript, PHP ve ASP.NET gibi dillere aşina olmak kaçınılmazdır.

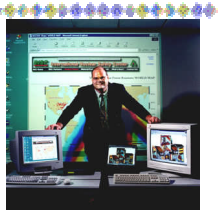
Eski bilgisayarlarda hâlâ eski kodların barınmasından dolayı, piyasada az kullanılan diller ile iyi kariyer yapabilirsiniz. Hatta bilen sayısı az olmasından dolayı, iyi ücret alabilirsiniz.

Çok Dil Kullanma

Bazı derleyiciler, kaynak kodu önce özel bir dosya türüne çevirir. "*Object file* – nesne dosyası" denilen bu dosya, kaynak dilden bağımsız hâle gelir. Farklı dilde oluşturulmuş nesne dosyaları, "*linker* - bağlayıcı" adı verilen programlar ile birleştirilir. Sonuçta ise çalıştırılabilir tek dosya oluşur.

Microsoft Windows, dillerin ortak kod kullanımı için DLL (*Dynamic Link Libraries* – Dinamik Bağ Kütüphanesi) yöntemi kullanır. Ara birimi olmayan bu dosyalar, farklı programlama dilleri kullanabilsin diye sisteme tanıtılmıştır.

Son bir yöntem, Microsoft'un .NET teknolojisidir. Aynı ara birimde ister C#, ister Basic veya başka bir dil ile program yazılabilir. Aynı veri tabanını ve çözümü ortak kullanarak birçok programcı bir arada çalışabilir. Her dilin kendine has avantajı kullanılarak, uygulamalar daha güçlü hâle gelebilir.



UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
1. Bilgisayarınızdaki temel imkânlar ile basit programlar yazınız.	En ilkel yöntemler ile program yapımını bulmaya çalışınız. ➢ "debug[††]" ile basit mikroişlemci komutları denenebilir. ➢ Not defteri ile basit bir web sayfası yapılabilir. ➢ *.bat yığın dosyaları ile sisteme yardımcı işlemler yapılabilir
2. C dili ve assembly dillerinde basit bir programın komutlarını karşılaştırınız.	Benzerlik, farklılık, iyi ve kötü yönlerini inceleyiniz.
3. Bilgisayarınızda yüklü olan görsel programlama dilini kullanarak basit bir uygulama hazırlayınız.	Görsel bir programlama dili ile metin tabanlı bir dilin benzerlik ve farklılıklarını inceleyiniz. Ara birimi hazırlarken işletim sisteminde yüklü programları örnek alabilirsiniz.
4. Web programlama dillerinden birini kullanarak basit bir sayfa hazırlayınız.	Web sayfalarında en çok kullanılan dilleri araştırınız. Ücretsiz hizmet veren sitelerden (geocities, brinkster ve tripod gibi) üyelik edinerek, site oluşturma denemesi yapabilirsiniz. Sitenizi arkadaşlarınız ile paylaşınız.

- Yapay Zekâ: Bilgisayarların filmlerde gördüğümüz gibi çalışmalarını sağlamaya çabalayan bilim dalı.
- *"Bilgisayarlar düşünebilir mi?"* sorusu *"Denizaltılar yüzme bilir mi?"* sorusuna benzer.
- Cebinde tornavida taşıyan programcılardan sakının...
- Kodu dokümante etmek mi? Neden *"kod"* dendiğini hiç düşündünüz mü?
- Programı dokümante etmeyin... Dokümanı programlayın.
- Hayır... Aklımı yitirmedim... Bir yerlerde yedeği olacak...
- Bir Mac'i hızlandırmak için en uygun ivme 9.8 m/sn^2 dir.
- Egr bn okyblyrsnz blgsr pgrmcs lbilrsnz.

[††] "Başlat*Çalıştır…*debug" komutu ile programı çalıştırabilirsiniz.

ÖLÇME VE DEĞERLENDİRME

OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; sonunda parantez olanlar doğru / yanlış sorularıdır. Verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Şıklı sorularda uygun şıkkı işaretleyiniz.

1. Bilgisayar çok akıllıdır, kullanıcının vereceği komutları tahmin ederek önceden cevabı hazırlar. ( )

2. Bilgisayar, bilgileri çözülemeyecek şekilde şifreleyebilir. ( )

3. Başkasının bilgisayarına, sitesine veya dosyalarına izinsiz olarak, ama zarar vermeden erişebiliriz. ( )

4. Bilgisayarlar kendi aralarında programlama dilleri ile anlaşırlar. ( )

5. Derlenmiş bir dosyadan istediğimiz dile geri dönüşüm yapabiliriz. ( )

6. Tüm dillerde bilgisayara zarar veren program yazılabilir. ( )

7. Aşağıdakilerden hangisi makine diline aittir:
A) 15
B) A
C) 0
D) %

8. Hangi dil, bilimsel matematik işlemleri yapmak için tasarlanmıştır?
A) Basic
B) Fortran
C) C
D) Pascal

9. Yazılım nedir?
A) Ana bellekte yer alan her türlü değer
B) Klavyeden girilen veriler
C) Bilgisayar programları
D) İşlemci

10. Hangisi yazılımdır?
A) Bilgi
B) Yazıcı
C) Derleyici
D) Klavye

11. Hangisi programın temel özelliklerinden olamaz?
A) Basitlik
B) Uyumluluk
C) Hatasız olmak
D) Doğruluk

12. Aşağıdakileri en kolaydan zor dile göre sıralanışı hangisidir?
1) Assembler 2) Delphi 3) C 4) Basic

A) 1-3-2-4
B) 3-1-4-2
C) 4-3-2-1
D)4-2-3-1

13. Hangi dilde Türkçe kod yazamayız?
A) Tupol
B) Pascal
C) Pozitif
D) Yunus

14. Aşağıdakilerden hangi eşleştirme hatalıdır (bilmediklerinizi araştırınız)?
A) AutoCAD - AutoLisp
B) 3DSMax - MaxScript
C) Corel - CorelScript
D) Office – VB

15. Aşağıdakilerden hangisi programcının asıl işi değildir?
A) Program yazmak
B) Program hatalarını gidermek
C) Programın pazarlanmasını sağlamak
D) Programın kurulum dosyasını yapmak

ÖĞRENME FAALİYETİ–2

AMAÇ

Bilgisayarda bir problemin çözüm aşamalarını yaparak, program yazmaya hazırlık yapabileceksiniz.

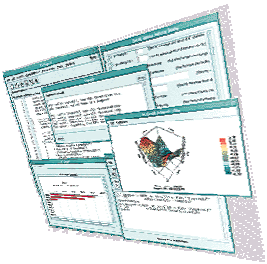
ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Günlük hayatta karşılaştığınız problemleri nasıl çözümlediğinizi dikkatlice gözlemleyiniz. Örneğin; okula geliş ve gidiş yolunda hangi taşıt araçları sıra ile kullandığınızı, hava ve yol durumuna göre çözüm olarak neler yaptığınızı anlatınız.
- Problemleri çözerken işlemleri belli bir sıra ile mi yapıyorsunuz yoksa çözüm için gerekenleri rastgele mi uyguluyorsunuz, buna dikkat ederek arkadaşlarınızla paylaşınız.
- "**Algoritma**" kelimesinin kaynağını araştırınız.
- Bir yemek tarifini araştırınız. Yemek yapımında izlenen adımları bir kâğıda yazınız. Alternatif durumları da belirtiniz.

Konularda bazı kelimelerin altı mavi dalgalı olarak çizilidir. Bu konuları araştırmanız tavsiye edilir.

2. PROGRAM YAZIMI



Şu anda oturup programlama yapabilir olsanız da, bu bir bebeğin önüne yemesi için gerekli malzemeyi savurmaya benzer.

Basit programları öğrenmek için hemen yazabilirsiniz. Ekranın ortasına okulunuzun adını yazmanız için uzun uzun plan yapmanıza gerek yoktur. Ama çok karmaşık programları, bilgisayarınıza aktarmadan önce, kâğıt üzerinde planlamanız gerekebilir.

Programı tasarlarken, ilk düşünmeniz gereken şey "hangi dili kullanacağınız" değildir. Programdan ne istediğinizi tam olarak bilirseniz, hangi dili kullanacağınıza kolay karar verirsiniz. Tek bir dili aramak yerine, değişik dilleri göz önünde bulundurunuz. Programınız çalıştığı sürece, hangi dil ile yapıldığı önemli değildir, özellikle de programınızı alan "müşteri" (kullanıcı da diyebiliriz) bununla ilgilenmez.

Bilgisayarda çözmeniz gereken problemin, sadece bir çözümü (algoritması) olmayabilir. Her çözümün kendi avantajı veya dezavantajı olabilir. Bazı algoritmalar daha

hızlı olarak işlemi bitirirken, aynı algoritmanın kod kısmını yazmak çok uzun sürebilir, mantığı çok karmaşık olabilir. Sık ağaçlarla kaplı bir ormana, dışarıdan baktığınızda içinden geçip gitmenin imkânsız gibi gözüktüğünü bilirsiniz. İşlemleri adımlara böldükçe hazmetmesinin daha kolay olduğunu göreceksiniz. Planlama sayesinde, yanlış program kodu üzerinde uğraşmaz, çözüm için zaman kaybetmezsiniz.

Programı tasarlarken şunları göz önünde bulundurmalısınız:

- **Kullanıcı**: programı kim kullanacak?
- **Hedef bilgisayar**: Windows mu, Linux mu, bir el bilgisayarı mı, yoksa bir süper bilgisayar mı?
- **Siz**: tek başınıza mı yapacaksınız? Takım halinde çalışacak iseniz hangi kısımları siz yazacaksınız?

Kullanıcı

Eğer programı sadece siz kullanacaksanız, görünümü istediğiniz gibi tasarlayabilirsiniz, çalışması yeterlidir. Satacak veya ücretsiz olarak verecek iseniz, karşınızdaki insanları tanımanız mantıklı olur. Kullanıcının bakış açısı ile bakarak programı tasarlamanız gereklidir.

Hedef bilgisayar

Kullanıcının kullandığı bilgisayarın türünü göz ardı edemeyiz. Programlama dilini buna göre seçeriz. Hangi donanımı kullandığı da önemlidir. Bir Mac kullanıcısı için bol hafıza, disk alanı, renkli grafikler, ses avantajı varken, bir el bilgisayarında ise daha az renk seçeneği, ekran çözünürlüğü ve belleği vardır. Bir program ne kadar çok çeşitli ortamda çalışıyor ise o kadar rağbet görür. Bu prensibe **"çapraz platform – *cross platform*"** denir. Kullanıcı sayısını artırma çabalarımız, beraberinde daha fazla hata türü ile karşılaşmamıza neden olur. Müşteri desteği sağlarken bunlara dikkat edilmelidir. Farklı sürümlere bölünmek, kârın bölünmesi demektir.

Not: Borland firması, görsel programlama dili Delphi'nin Linux sürümü olan Kylix'i çıkarmıştır. Fakat sadece 1.0 sürümünde kalakalmış, proje ileri götürülememiştir.

Siz

Herhangi bir programlama dili ile uğraşırken, önce kendi yeteneklerinize dikkat edersiniz. İyi bir proje aklınıza gelmiş olabilir. Ama deneyiminiz az ve bilginiz az olunca, kendi programınızı yazmak zor gelebilir. Acemi biri için C ve C++ ile program yapımı çok zaman alıcı olabilir. Daha basit olarak Visual Basic'te program yazmayı öğrenip, piyasada çalışmaya başlayabilirsiniz.

Bir günde program yazan çok az insan vardır. Program kod yazımı zaman alır, acele yapıldığında içi hatalarla dolabilir. Ne yapacağınızdan emin olduktan ve istenen programın ne iş yapacağını anladıktan sonra program yapımına başlayınız.

Altın Kelepçeler

Birçok insan program yazmak yerine "programcı" işe alırlar. Aman dikkat! Özgür çalışan bazı programcılar "altın kelepçe" kuralı ile geçiniyor olabilir. Bir programcıyı işe alırsınız, biraz para verirsiniz. Programcı tam istediğiniz gibi bir program yazmaz. Parayı kaybetmek yerine, biraz daha para vererek, programın yeni sürümünü yaptırırsınız. İşte bu noktada kelepçelendiniz demektir. Programcıya devamlı para verir ama hiçbir zaman istediğiniz programı elde edemezsiniz. Başka programcıya da geçemezsiniz; çünkü kaynak kod elinizde değildir. Elinizde olsa bile kimse o programı değiştirmek için zaman harcamaz.

2.1. Prototip (Kalıp) Oluşturma

Tam çalışmayan veya yanlış çalışan bir program olmaması için programa aylarca zaman harcamadan önce, programcılar mimarların maket modeller yapması gibi, "prototip" oluştururlar. Bunu normal bir program gibi, menüleri, formları, diyalog kutuları olan ama içinde kod olmayan bir uygulama olarak düşünebilirsiniz. Fikir vermek amacı ile yapılırlar. Prototipten memnun kalınca, programcı kodlamaya geçer. Birçok programcı Visual Basic veya Delphi gibi görsel bir programlama dili ile prototip hazırlar.

2.2. Sahte Kod ve Algoritma Yazma

Programlama dili seçseniz bile, hemen kod yazımına geçemezsiniz. "*Pseudocode* – sahte kod" denilen kodlama ile programın işlemleri yazılır. Sahte kod kurduğunuz mantıktaki kaçakları bulmaya yarar.

Örneğin; eğer bir şehre füze göndermek isterseniz adımlarınız şu şekilde olabilir:[##]

1. Hedefin koordinatlarını al
2. Füzenin şu anki koordinatını al
3. Hedefi vurması için gerekeni hesapla
4. Füzeyi ateşle

1. adımın daha da gelişmiş hâli şöyle olabilir:

1.a. Füze teknisyeninden hedef koordinatları edin
1.b. Hedef koordinatların geçerli olduğundan emin ol
1.c. Hedef koordinatları hafızaya kaydet

1.a. maddesini daha alt basamaklara bölebiliriz.

1.a.i. Koordinattan emin ol
1.a.ii. Hedef füzenin atış uzaklığı içinde mi, kontrol et
1.a.iii. Hedef dost bölge mi, kontrol et

Bu **"üstten aşağı – *top down*"** tasarım ile kaba bir taslak gittikçe ayrıntılı şekilde parçalanır. Genelden özele doğru işlemleri belirleriz. Bu zaman kaybı gibi görülebilir. Plansız olarak programınızı yazmaya başlayabilirsiniz. Ama bu arabaya atlayıp Samsun'dan Mersin'e gitmek için kuzeye doğru gitmeye benzer.

[##] İlk üretilen bilgisayarlardan istenen, hızlı hesap gücü sayesinde doğru bir şekilde sonuç vererek, silahlara destek olmaktı.

Sahte kodlar programınızın ana hatlarını belirtir. Böylece bilgisayarın başaracağı iş için gerekli tüm veriyi toplamış olursunuz.

Büyük bir programın temel adımlarının olduğu algoritma **"makro seviye"** olarak isimlendirilirken, detayların algoritmalarına **"mikro seviye"** denir. Makro seviye içinde birçok mikro seviye algoritma vardır. İlk kez program yazan kişiler detay kısımlarını genelde görmezden gelirler. Üzerinde çok düşünülmeden yazılan program, her ihtimale uygun cevap veremeyeceği için hatalı çalışır.

Eğer plansız olarak program yazarsanız, kaçınılmaz son, kısa süre sonra "kod çöplüğü" olabilir. Programı alan kişi nasıl kod yazdığınızı umursamasa da, böceklerle (*bug*)* dolu bir yazılımı, siz olsanız kullanır mıydınız?

Temelde programınızı üç ana bölüme ayırabiliriz:
Giriş: Bilgisayarın üzerinde çalışacağı veri kullanıcı tarafından girilir.
İşlem: Bilgisayar girilen bilgiyi işler.
Çıkış: Bilgisayar insanların anlayacağı şekilde ekrana sonucu gösterir.

Örnek:
Giriş:
a. Ürünün adını al.
b. Ürünün miktarını al.
c. Bir dosyadan ürünün fiyatını oku.

İşlem:
a. Ürünün toplam fiyatını hesapla.
b. Gerekli indirimi yap.

Çıkış:
a. Toplam satış değerini yaz.

Örnekteki adımları "sahte kod" olarak alt alta toplarsak:
1. Başla.
2. Ürünün adını al.
3. Ürünün miktarını al.
4. Bir dosyadan ürünün fiyatını oku.
5. Ürünün toplam fiyatını hesapla.
6. Gerekli indirimi yap.
7. Toplam satış değerini yaz.
8. Bitir.

Bazı temel sahte kod komutları şunlardır:

Başla … Bitir : Sahte kodun ilk satırı başlangıç ve son satırı bitişi için yazılır.
Oku : Kullanıcı girişi için yazılır.
Yaz : Kullanıcıya bilgi veya sonuç göstermek için yazılır.
Eğer İse Değilse : Şartlara göre akışın değişmesinde kullanılır.

* Böcek (bug): Programda hatalara sebep olan program satırlardır.

Örnek:

1. Başla
2. Sayısal sayı1, sayı2, sonuc
3. Yaz; "İki sayı giriniz"
4. Oku; sayı1
5. Oku; sayı2
6. sonuc = sayı1 + sayı2
7. Yaz; "Sonuç: " & sonuc
8. Bitir

Not: Birden fazla değer peş peşe okumada, kısaca şu şekilde de tek satırda okuma yapabilirsiniz:

Yaz; "İki sayı giriniz"
Oku; sayı1, sayı2

Sahte kodu satır numaraları olmadan "**bloklar**" hâlinde yapabiliriz. Her blok açıldığında biraz daha içeriden yazmaya dikkat ediniz. Böylece program daha okunaklı hâle gelir.

Örnekler:

➢ Öğrenci notları ile ilgili sahte kod yazalım:

```
Başla
     Sayısal öğrenciNumarası
     Yaz; "Öğrenci numarasını giriniz"
     Oku; öğrenciNumarası
     Veritabanından öğrenciNotu bilgisini oku
     Eğer öğrenciNotu 50'den büyük İse
               Yaz; öğrenciNumarası & " Geçti"
     Değilse
               Yaz; öğrenciNumarası & " Kaldı"
     Eğer Bitti
Bitir
```

➢ 2 sayıyı karşılaştıran sahte kod yazalım:

```
Başla
     Sayısal Sayı1, Sayı2
     Oku; "Birinci sayıyı giriniz", Sayı1
     Oku; "İkinci sayıyı giriniz", Sayı2
     Eğer Sayı1 Sayı2'den büyük İse
          Yaz; "Sayı1 Sayı2'den büyüktür."
               Eğer Sayı2 Sayı1'den büyük İse
                    Yaz; "Sayı2 Sayı1'den büyüktür."
               Değilse
                    Yaz; "Sayı1 ve Sayı2 eşittir."
               Eğer Bitti
     Eğer Bitti
Bitir
```

➢ Kitap indirimi ile ilgili sahte kod yazalım:

```
Başla
     Yaz; "Kitap türünü giriniz"
     Oku; kategori
     Eğer kategori A İse Yaz; "A kategorisinde %10 indirim vardır."
     Eğer kategori B İse Yaz; "B kategorisinde indirim yoktur."
Bitir
```

❓ Siz de basit olarak bir çay makinesinin, ütünün ve telefonun kullanımı ile ilgili adımları yazınız.

2.3. Akış Şemaları

Akış diyagramları görsel olarak basitçe bir problemin akışının gösterimidir. Microsoft firmasının Visio, Word, Powerpoint gibi birçok program ile bu grafikleri hazırlayabilirsiniz. Biz de bazı örnekleri denemek ve çizmek için "Flow Chart Visual Programming Language[§§]" isimli açık kaynak kodlu[***] bir proje olan programı kullanacağız. Ayrıca "Crocodile Technology" isimli program ile de akış şemalarının hem tasarlanması hem de çalıştırılması imkânımız var.

Şema ile program hazırlığı ayrıca UML ile yapılabilir. UML[†††] İngilizce *Unified Modelling Language* (Birleştirilmiş Modelleme Dili) kelimelerinin baş harflerinden meydana gelir. Modelleme sırasında kullanılacak bir dizi şematik gösterimi teşkil eder. Genelde nesne tabanlı sistemlerin analiz ve modelleme aşamalarında kullanılır. Nesnelerin birbirleri arasındaki ilişkilerini ve kendi iç yapılarını gösterir. Programlama dilinden ve işletim sisteminden bağımsız bir modelleme dilidir. "IBM Rational Rose" programı ve ArgoUML ile tasarım yapabilirsiniz.

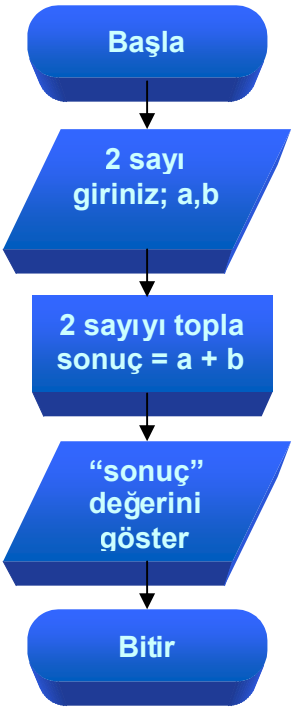
ANSI (*American National Standards Institute*), akış diyagramları ile ilgili bazı tavsiyelerde bulunmuştur:

- Akış diyagramındaki standart sembollerle programın mantığı sunulmalıdır.
- Akış diyagramı basit, açık ve kolay takip edilebilir olmalıdır.
- Her akış diyagramının bir adet başlangıç ve bitiş noktası bulunmalıdır.
- Akış diyagramının takip yönü yukarıdan aşağıya, soldan sağa doğru olmalıdır.
- Veri girişleri mantıklı bir sırada listelenmelidir.
- Başlangıç ve bitiş sembollerinde tek hat kullanılmalıdır.
- Akış diyagramı çok karmaşık hâle gelince, bağlantı noktaları ve alt program kullanılmalıdır.
- Test verisi ile şema denenmelidir. Değerleri bir tabloya aktarılmalıdır.

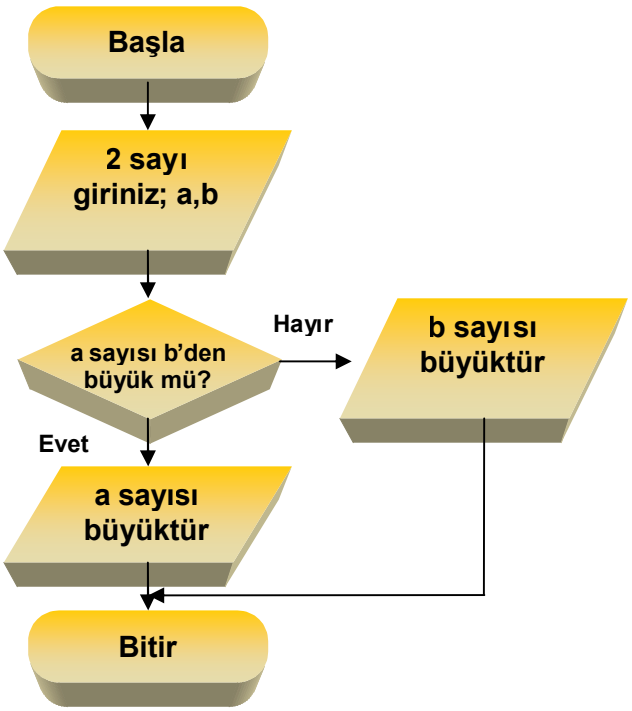
[§§] 2005 yılında 3.0 sürümüne ulaşan "Visual Basic 6" ile yapılmış programıdır.
[***] Programın dağıtımında kaynakları da dâhil olursa "açık kaynak kod" proje adı verilir.
[†††] Bir standart olarak "Object Management Group" (OMG) tarafından dünyaya yayılmaktadır.

Şekil	Görev
	Giriş veya çıkış (*input* ve *output*) Kullanıcı etkileşimi için gereklidirler
	İşlem, hesaplamalar (*process*, *calculation*) Bilgisayarın asıl yaptığı işler burada gerçekleşir
	Karar, eğer (*decision*, *if*) Program iki ihtimale göre akışını değiştirir, her ihtimalin ne işlem yapacağını göstermeyi unutmayınız
	Alt program, fonksiyon (*sub program*, *function*) Akış diyagramı karmaşık hale gelince, tekrarlanan belli kısımlarını alt program haline getirebilirsiniz
	Akış çizgisi (*flowline*) Genellikle yukarı ve sola doğru çizilmezler
	Başla ve ya Bitir (*Start* ve *Stop*) Her programın bir başlangıç ve bir bitişi vardır
	Birleştirici, bağlantı noktası (*connector*) Birçok sayfadan oluşan akış şemalarını birbirine bağlar
	Döngü (*for*, *cycle*) Başlangıç, artım ve sonlanma şartlarını iyi belirleyiniz
	Dosya (*file*, *storage*) Saklanması gereken bilgileri kaydederiz

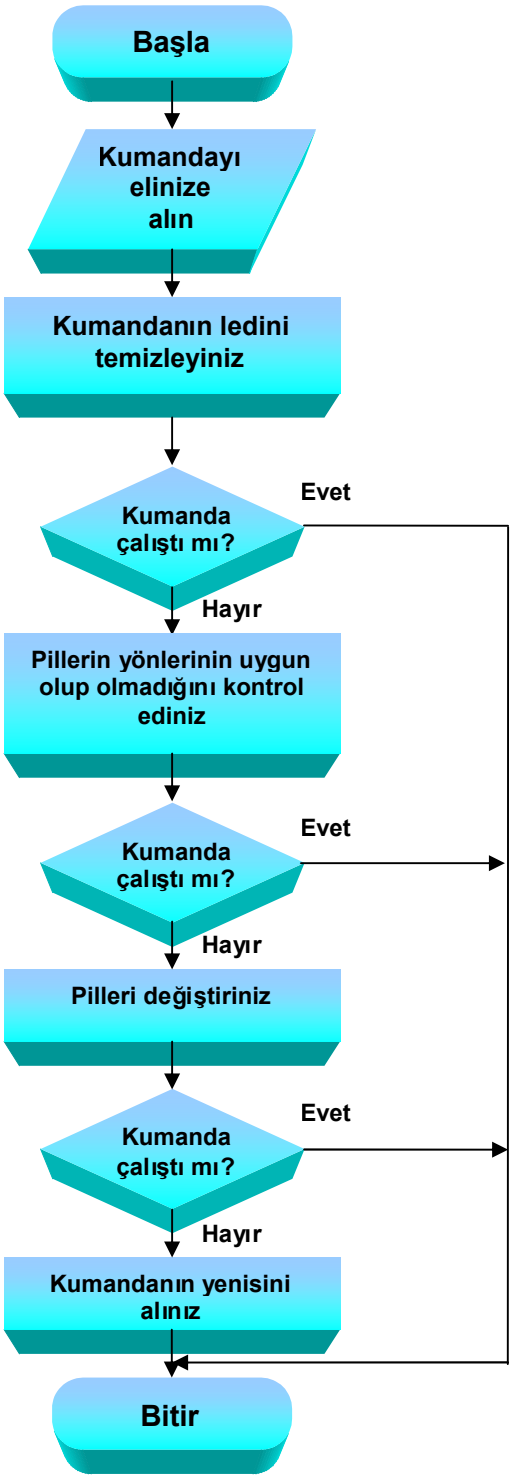
İki sayıyı toplayıp sonucunu gösteren programın akış şeması şu şekildedir:



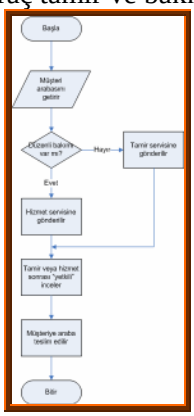
İki sayıyı karşılaştırma programının akış şeması şu şekildedir:

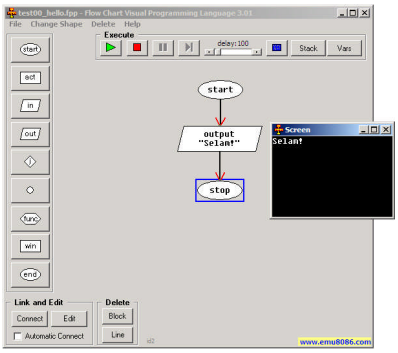


Bozuk bir televizyon kumandası ile ilgili çözüm akış şeması şu şekildedir:



Araç tamir ve bakımı ile ilgili bir örnek akış şeması:





Resim 2.1: Görsel olarak akış diyagramı hazırlama ve çalıştırma örneği

Akış diyagramları sahte kodlara göre daha anlaşılır ve etkindir. Geliştirme aşamasında iyi bir yol göstericidir, hatalar daha rahat görülür.

Dezavantajları şunlardır:

- Çok sayfa süren akış diyagramları zor takip edilir.
- Bir grafik programı ile çizimi zaman kaybına neden olur.
- Bir değişiklik yapmak gerekince şemanın tekrar çizilmesi gereklidir.

2.4. Karar Tabloları

Başka bir programlama yardımcısı karar tablolarıdır (*decision table*). Çok karmaşık karar yapıları bu sayede kolayca çözülebilir hâle gelir.

Şart kütüğü, şart, durum kütüğü ve durum olmak üzere tablo halinde 4 parçadan oluşur.

Şart kütüğü	Şartlar
Durum kütüğü	Durumlar

Şart kütüğü: Hangi kararlar verilecek?
Şartlar: Şartın alternatifleri neler?
Durum kütüğü: Belli şartta ne yapılacak?
Durumlar: **E** (Evet) ve **H** (Hayır) halinde hareket seçenekleri

Örnek:

ABC şirketi bir sekreter eleman alacaktır. Şartları:

a. Aday bayan olmalı
b. En az 2 yıl deneyimli olmalı
c. Bekâr olmalı

ABC Firmasının Eleman İhtiyaç Karar Tablosu								
Şart Kütüğü	**Şartlar**							
Bayan mı?	H	E	H	E	H	E	H	E
2 yıl deneyimli mi?	H	H	E	E	H	H	E	E
Bekâr mı?	H	H	H	H	E	E	E	E
Durum Kütüğü	**Durumlar**							
Kabul durumları	H	H	H	H	H	H	H	**E**

Dikkat ettiyseniz en fazla sekiz durum olabiliyor. Bu sayı 2 ^ 3 = 8* hesabından geliyor. İkilik sayı sistemindeki gibi sadece iki ihtimal (E / H) olduğu için taban ikidir. Üç adet de limit sağlayan şartımız var.

Tüm olabilecek durum gözden geçirilerek hata olması engellenir. Akış diyagramına göre daha okunaklıdır. Fakat ilk kez program yazanlar genellikle akış diyagramı çizmeyi tercih ederler.

? Aşağıdaki soruların sahte kod, akış şeması veya karar tabloları yöntemi ile çözümünü yapınız:

1. Bir sınıftaki 5 öğrencinin notları girilir, ortalaması hesaplanır.
2. Bir elektronik mağazasında 37 ekran televizyonlar %20, 54 ekran televizyonlar da %15 indirim yapılıyor. Buna göre ekran boyutu girildiğinde, indirim oranını mesaj olarak gösteriniz.
3. Bir üniversite teorik dersin 25 saatine 1 kredi, laboratuvarın 10 saatine 1 kredi veriyor. Ders ve laboratuvar saati girilince toplam krediyi hesaplayan programı yazınız.

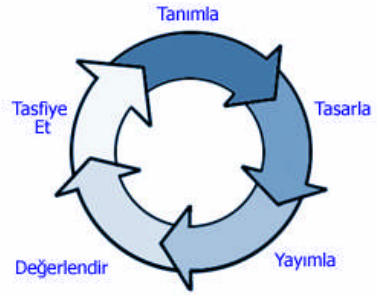
? Sahte kodu karışık olarak verilen problemin satır sıralarını düzeltiniz. Problem şudur; 3 ürünün satış ve maliyet fiyatları girilir, kârları toplanır ve ortalamaları alınır.

Üç ürünün kârını topla.
Üçüncü ürünün satış fiyatını gir.
Birinci ürünün satış fiyatını gir.
Toplamı 3'e bölerek ortalamasını bul.
Üçüncü ürünün maliyet fiyatını gir.
Bitir.
İkinci ürünün satış fiyatından maliyet fiyatını çıkarıp kârını bul.
İkinci ürünün maliyet fiyatını gir.
Ortalama kârı göster.
Başla.
Üçüncü ürünün satış fiyatından maliyet fiyatını çıkarıp kârını bul.
İkinci ürünün satış fiyatını gir.
Birinci ürünün satış fiyatından maliyet fiyatını çıkarıp kârını bul.
Birinci ürünün satış fiyatını gir.

* Buradaki ^ şapka karakteri Basic dilinde "**üs**" alma işlemi için kullanılan işleçtir.

2.5. Bir Programın Hayat Döngüsü

Çok az program; yazılır, geliştirilir ve öylece bırakılır. Bunun yerine, kullanılmayacak hale gelene dek, program değişik döngülerde güncellenir. Fakat her şey devamlı değişmez, mesela neredeyse her yıl yeni bir kelime işlemci programı sürümü çıkarılmasına rağmen, alfabedeki harfler hep aynı kalır.



Genel olarak tipik bir program döngüsü şöyledir:

- **Geliştirilme aşaması** (ilk oluşturulma ve yayımlanma)
- **Bakım döngüsü** (olabildiğince kısa zamanda böceklerin ayıklanması)
- **İyileştirme döngüsü** (yeni özellikler katma)

Geliştirme aşaması: Her program programcının bilgisayarında boş bir ekran olarak başlar. Bir fikirden, gerçek bir program oluncaya dek, geliştirme aşaması devam eder.

1. Fikir doğar.
2. Programın tipik hali ortaya çıkar.
3. Hangi bilgisayarlarda çalışacağı belirlenir.
4. Sahte kodlar yazılır.
5. Bir veya daha fazla dil seçilir.
6. Program yazılır.
7. Test edilir. (**Alfa** sürümü denir.)
8. Problemler giderilir. (7 ve 8. adım gerektiği kadar tekrar edilmelidir.)
9. Diğer insanlara test etmeleri için verilir. (**Beta** sürümü denir.)
10. Problemler giderilir. (9 ve 10. adım gerektiği kadar tekrar edilmelidir.)
11. Halka program sunulur ve pazarlandığı gibi çalışması umulur.

Bakım döngüsü: Çoğu programcı, eski bir programı güncellemek yerine yeniden program yazarlar. Güncellemek aynen, kendi evinizi temizlemek yerine, başkasının evini temizlemek gibidir.

Bakım gerektiğinde şu adımlar yapılır:

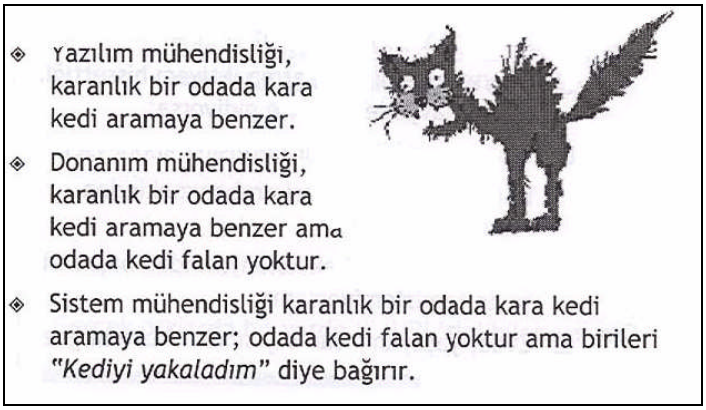
1. Programın hangi kısmında hata olduğu bulunur.
2. Böcek temizlenir.
3. Yeni böceklere sebep olunup olunmadığı anlaşılması için test edilir.
4. Testte görülen böcekler temizlenir. (1 ve 4 arası her böcek için tekrar edilir)
5. Yazılım için yama geliştirilir.

İyileştirme döngüsü: Yazılımın tamiri ve kararlı, güvenilir hâle gelmesi, ücretsiz bir destek hizmeti olduğu için şirket para kazanmaz. Bunun yerine yeni eklentilere sahip sürümleri satmak kazandırır.

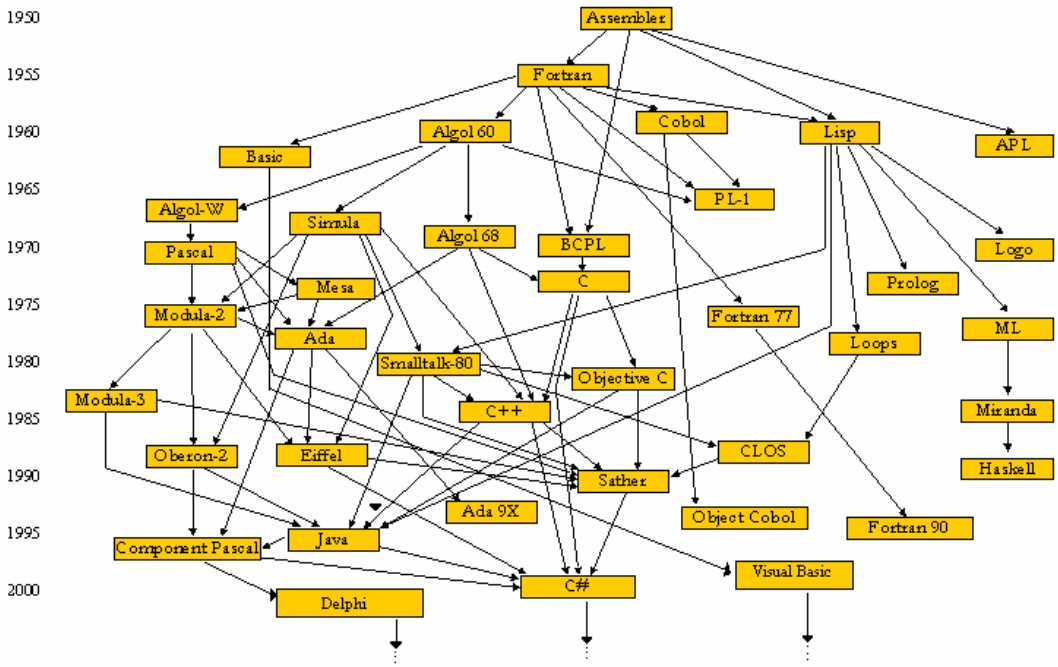
Yeni yazılım ve donanım ortamlarının avantajını kullanmak için birçok program, tekrar elden geçirilir.

1. Programa ne eklenecekse belirlenir.
2. Bu özellik nasıl çalışacak ise sahte kodlar ile yazımı yapılır
3. Program değiştirilir.
4. Alfa sürümü olarak test edilir.
5. Hatalar tespit edilip giderilir.
6. Beta sürümü olarak test edilir.
7. Beta testçilerinin hata raporları okunur. (1 ve 7. adımlar arası her eklentide tekrar edilir.)
8. Halka sunulur ve çıkabilecek yeni hatalar beklenir. (Bakım döngüsüne girilir.)

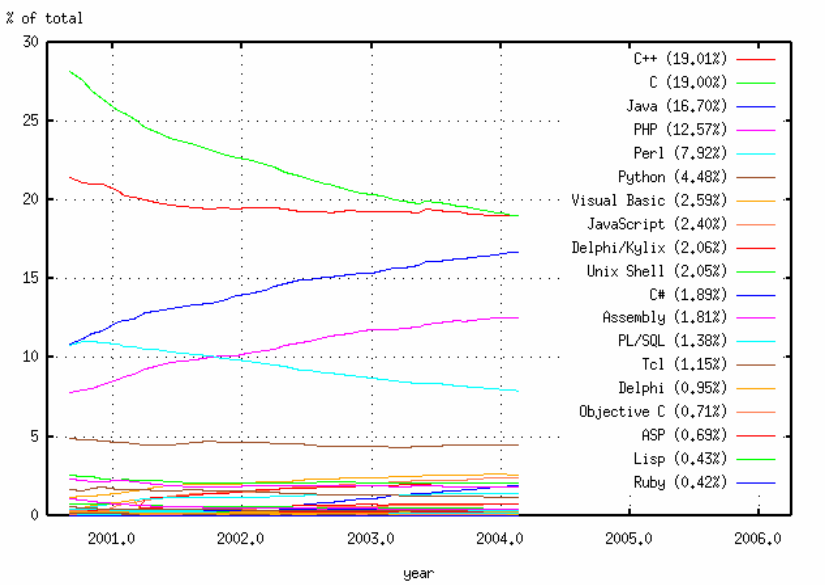
Üniversitelerdeki ve sektördeki "yazılım mühendisliği" sözündeki mühendislik, aslında daha çok "**sanat**" olarak kullanılabilirdi. Yazmak, değiştirmek ve güncellemek yüksek bir zekâ ve matematik bilgisi gerektirmez. Daha çok üretkenlik, azim ve bol hayal gücü gerektirir.



İstediğiniz şekilde program yazabilirsiniz. Ama ileride olabilecek problemleri önlemek için en iyi yol, **planlı** olmaktır.



Resim 2.2: Programlama dillerinin soy ağacı



Resim 2.3: Programlama dillerinin kullanım oranları

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
1. Problemi tespit ediniz.	Günlük hayattan bir örnek bulunuz.
2. Problem ile ilgili verileri belirleyiniz.	Verilen bilgileri ve istenenleri ayrı ayrı not alınız.
3. Algoritma adımlarını belirleyiniz.	Adımları belirlerken gereksiz adımlardan kaçınınız. Her adımı sadece bir işlemi temsil edecek şekilde hazırlayınız.
4. Her adıma bir numara veriniz.	Her algoritmanın başının ve sonunun belli olması gerektiğini unutmayınız.
5. Algoritmayı kontrol ediniz.	Algoritma adımlarında problemin çözümünü etkileyecek herhangi bir belirsizlik olmamalıdır. Algoritma hazırlanırken eğer bir karar yapısı varsa; yapılan işlem sonucunda her türlü alternatifi değerlendiriniz. Bir sorunun cevabının "Evet" olabileceği gibi "Hayır" olabileceğini de unutmayınız.

ÖLÇME VE DEĞERLENDİRME

OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; sonunda parantez olanlar doğru / yanlış sorularıdır. Verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız.

1. Akış diyagramı; algoritmanın şekillerle ifade edilmiş hâlidir. ( )

2. Bir akış sembollü içinde aynı anda sadece bir işlem gerçekleştirebilir. ( )

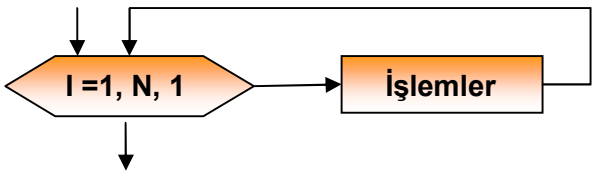
3. Ayrı hazırlanan iki akış diyagramını birleştirmek için → işareti kullanılır. ( )

4. Algoritmada işlem sırası önemli değildir.

5. Kullanılan her veri hemen en üstte tanımlanmalıdır. ( )

6. Bir karşılaştırma işleminin sonucu "Evet" ya da "Hayır" gibi bir ifadedir. ( )

7. Aşağıda verilen akış diyagramındaki ifadenin anlamı aşağıdakilerden hangisidir?

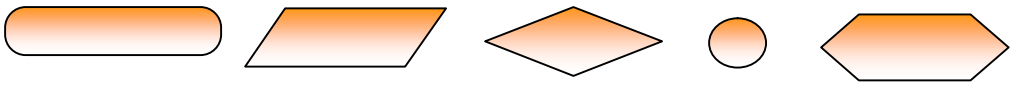


A) I değeri eğer N'ye eşit ise 1 artır, işlemleri yap
B) I değerini N'den başlayarak 1'er 1'er artır, işlemleri yap ve çık
C) I değeri 1'e eşit ise N'ye 1 ekle, işlemleri yap
D) I'yı 1'den başlayarak N'ye kadar 1'er 1'er artır. I'nın her değeri için işlemleri yap

8. Aşağıdakilerden hangisi bir problem olabilir?
A) Piyangodan büyük ikramiye çıkmasını beklerken amorti çıkması
B) Piyangodan büyük ikramiye çıkmasını beklerken büyük ikramiye çıkması
C) Arasına piyango bileti konulan kitabın yerinin bilinmemesi
D) Piyango çekilişinin her zamanki yerinde ve zamanında yapılması

9. Aşağıda akış diyagramı sembolleri ve anlamları verilmiştir.

1. 2. 3. 4. 5.



6.

( ) Başla / Dur ( ) Bilgi Girişi ( ) İşlem ( ) Döngü
( ) Karar ( ) Bağlantı

Yukarıdaki verilere göre aşağıdaki seçeneklerden hangisi <u>yanlıştır?</u>

	Sembol Nu.	Anlamı
A)	1	İşlem
	5	(Bağlantı
B)	3	Karar
	6	Bilgi girişi
C)	2	Karar
	6	Bağlantı
D)	3	Karar
	5	Döngü

ÖĞRENME FAALİYETİ–3

AMAÇ

Programlama dillerindeki yazım araçlarını kullanabileceksiniz.

ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Bilgisayarınızda kurulu olan bir programın (işletim sistemi de olabilir) kurulum dosyalarını ve kurulum adımlarını inceleyerek sınıfınızda sunu olarak tanıtınız.
- En rahat program yazabileceğiniz yardımcı araçları araştırınız. Programın hangi programlama dillerini desteklediğini inceleyiniz.

Konularda bazı kelimelerin altı mavi dalgalı olarak çizilidir. Bu konuları araştırmanız tavsiye edilir.

3. PROGRAMLAMA ARAÇLARI

Programlamayı kolaylaştırıcı birçok araç vardır. Bu araçların kullanımını öğrendikten sonra program yazmak için fazla zaman kaybetmezsiniz.

Temel gereksinimler:

- **Düzenleyici (*editor*):** Komutları yazmamız için kullanılır.
- **Derleyici (*compiler*):** Komutlarımızı makine diline çevirir.
- **Böcek ayıklayıcı (*debugger*):** Programınızdaki hataları bulmaya yardımcı olur.

Ek olarak:

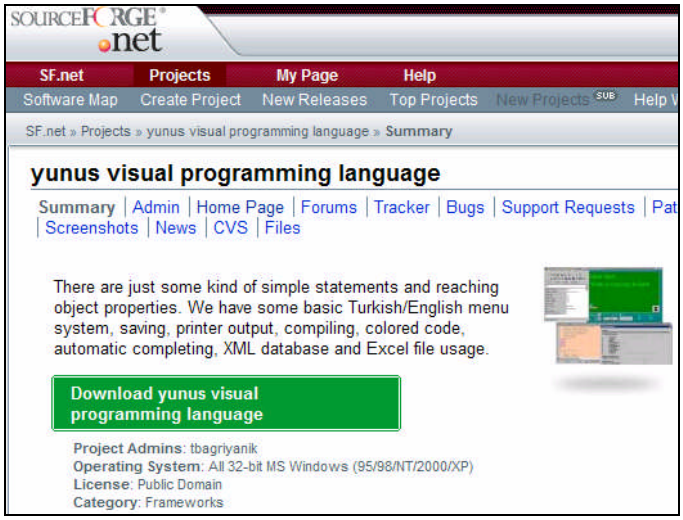
- **Yardım dosyası hazırlayıcı:** Kullanıcıya kılavuz olacak el kitabı şeklinde, yardım sağlamak için kullanılan programlardır.
- **Kurulum programı (*setup* / *installation*):** Programın dosyalarını tek dosya haline getirerek, diğer uyumlu bilgisayarlara kurmaya yarar.

Linux ve Açık Kaynak Hareketi

İlk zamanlarda programcılar gönüllü olarak kaynaklarını paylaşarak, programların gelişmesine katkıda bulunma fikrindeydiler. Zamanla, emeklerinin karşılığında para kazanmak istediler. Kaynaklarını kendilerine sakladılar. Böylece bir "**yazılım endüstrisi**" meydana geldi. Yayım tarihleri aksayan, güvensiz ve kötü yazılmış yazılımlar da arttı.

Fakat kaynak kod paylaşımı fikri tekrar ortaya çıkarak, Linux

işletim sisteminin ortaya çıkmasına sebep oldu. Para vermeden kullanılabiliyor ve kaynak kodlara göz atabiliyorsunuz.



Resim 3.1: Açık kaynak proje sitesi örneği

*Not: Bazı diller yorumlayıcı (interpreter) olarak çalışırlar. Yorumlayıcı, derleyici gibi *.exe (veya çalıştırılabilir dosya) dosyayı diske değil, satır satır geçici olarak hafızaya oluşturarak programı çalıştırır.*

3.1. Düzenleyici

ASCII (*American Standard Code for Information Interchange*) dosya olarak yazabileceğiniz tüm programlar işinizi görür. Mesela not defteri olabilir. Metin dosyaları tüm bilgisayarlarda standart olan bir dosya türüdür. Ama bir kelime işlemci programı kullanmak yerine programcılar kendi düzenleyicilerini geliştimişlerdir. Programı, yazması, düzenlemesi ve yazdırılması kolay bir araç gibi düşünebilirsiniz.

3.2. Derleyici

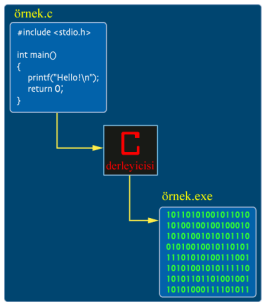
Derleyici kaynak kodu makine diline çevirir. Eğer yazım hataları yok ise, sonuçta çalıştırılabilir bir dosya meydana gelir. Bu dosyayı isterseniz diğer insanlara verebilir veya satabilirsiniz. Mesela Microsoft Word programı derlenmiş bir dosyadır.

Kaynak kod değişmediği sürece tekrar derleme işlemi yapılmasına gerek yoktur. Derleyici işlemci türünüze göre kod oluşturur. Eğer işlemcinizin mimarisi farklı ise o işlemciye göre tekrar derlenmesi gereklidir.

Perl ve Visual Basic yorumlayıcı (*interpreter*) olarak çalışır. Kodlar satır olarak çalıştırılır. Çoğu web programlama dili yorumlanarak çalıştırılırlar. Mesela web tarayıcıları, sitedeki JavaScript kodlarını yorumlarlar. Yani ortada derlemiş bir dosya yoktur. Bu da sitedeki kaynak kodun bilgisayara metin olarak indirilip, çalıştırılması anlamına gelir.

P-code (sanal makine kodu) denen hem derleyici hem de yorumlayıcı programlama dilleri de vardır. Birçok bilgisayarda az değişiklik yapılarak çalışabilen program yazmak için programcılar, p-code programlarını tasarlamışlardır. Program sanal makine koduna dönüştürülür. P-code olarak derlenmiş kodu çalıştırmak için bir yorumlayıcı kullanılır. Java bu şekilde platformdan bağımsız çalıştırılıyor. Bu yöntemle kaynak kod korunmuş olur.

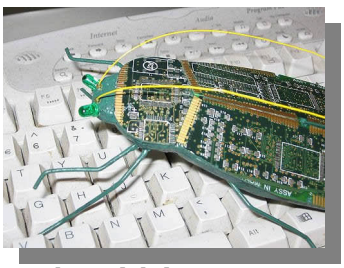
Aslında programlar her ne kadar derlenme sonucunda korunsa da, "***decompiler*** – kaynak koda dönüştürücü" ile makine dilinden dönüştürülebiliyor. P-code tamamen orijinal koduna "*decompile*" edilebilir, diğer dillerde de bu vardır. Fakat tamamen orijinal kod haline gelemez. Hangi üst seviye dille yazıldığını bilmek mümkün olmadığı için, makine dili kodları, assembly diline dönüştürülebilir. Bu kodu alarak kendiniz kaynak kodları yazabilirsiniz. Ya da en azından o program ile ilgili fikir alabilirsiniz.



Resim 3.2: Bir C programının derlenmesi

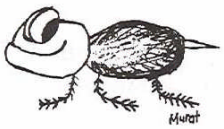
Not: AMD ve intel aynı x86 (CISC) kod ailesindendir, IBM firmasının PowerPC'si ise farklı mimaride (RISC) üretildiği için kodları ayrıca derlenmelidir. Ayrıca 32 bit ve 64 bit işlemcilerde assembly kodu farklı olduğu program tekrar derlenmelidir.

3.3. Böcek Ayıklayıcı (Debugger)



Çok az program **%100** tam çalışır. Hatalar yüzünden programlar çöker, veriler kaybolur, ya da program tuhaf çalışır. Aslında program her zaman aynı durumda aynı sonucu vermelidir. Ama yazılım ve donanım farklılıklarından dolayı her zaman istenen sonuç elde edilemeyebilir.

Yazılımın olduğu her yerde böcek vardır. Derleyiciniz, işletim sisteminiz ve tabii ki kendi programınızda da böcekler olabilir. Maalesef bir böceği temizlemek yeni böcekler oluşmasına sebep olabilir.



1945 yılının Ağustos ayında Harvard'daki MARK I bilgisayarında çalışanlar bir şeylerin garip gittiğini farkettiler.

Program beklendiği gibi çalışmıyordu.

Uzun araştırmalardan sonra sorunun bilgisayarın röleleri arasına sıkışmış olan bir güveden kaynaklandığı anlaşıldı.

COBOL dilinin tasarımını yapan ekipten Tuğamiral Bayan Grace Hooper'ın (1906 - 1992) anlattığına göre ilk "debug" etme işi bir cımbızla halledilmişti. O gün bugündür bilgisayarda işler iyi gitmeyince ilk akla gelen hep böcekleri suçlamak olmuştur.

MARK I'i çökerten bu ilk "bug", halen Naval Surface Weapons kolleksiyonunda yer alan log defterlerinin bir sayfasına yapıştırılmış olarak saklanmaktadır.

Yazdığınız programlarda böcek* ayıklamanın (***debugging***) yolları; adım adım çalıştırmak (***step***), durak noktaları kullanmak (***breakpoint***), değişkenleri izlemektir (***watch***).

Adımlamak: Programın satır satır çalıştırılmasıdır. Böylece böcek olan satır incelenebilir. Neyi yanlış yaptığınızı yazdığınız kodları inceleyerek öğrenirsiniz.

Durak noktaları: Tüm programı satır satır çalıştırmak yerine, şüphelendiğiniz yerlere durak noktaları ekleyerek, programın oraya kadar çalışıp durması sağlanır.. Program durunca da satır satır çalıştırılarak nerede hata olduğu bulunmaya çalışılır.

Değişken izleme: Program çalışırken, hafızadaki değişkenlerin değerlerinin nasıl değiştiğini görebilirsiniz. Yanlış bilgi var ise, programdaki hatayı anlamaya, değişkenin nereden etkilendiğini bulmaya çalışırsınız.

3.4. Yardım ve Kurulum

Yardım dosyası ve kurulum hazırlama programları ayrıca satılır veya ücretsiz olarak internette bulunabilir. Kurduğunuz programlama dilinde basit olarak yardım hazırlamak için yardımcı program olabilir. Çoğu programın **yardım dosyası** vardır. İlk kez programınızı kullanan veya bir konuda takılan biri için ilk akla gelen şey, "Bu programın yardımı nerede?" sorusudur. Yardım dosyası olan programlar daha modern hale gelir.

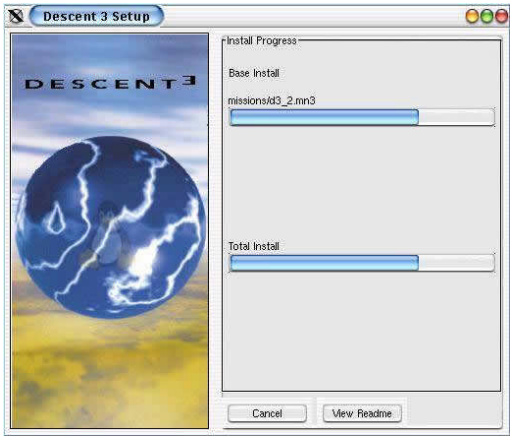
Program test edildikten, hataları giderildikten ve yardım dosyası hazırlandıktan sonra son adım olarak **kuruluma** geliriz. CD veya disketlere bölünecek şekilde "**kur**" dosyası yapılır.

Not: Bazı programlar doğrudan istenen herhangi bir dizinde çalışsa bile, genellikle ek dosyalar (veritabanı dosyaları, DLL, OCX dosyaları gibi) problem çıkarabilir.

---

* İlk bilgisayar böceği ile ilgili resmi araştırınız

Kurulumu olmayan programları kullanıcı denemeye bile çalışmayabilir. Çünkü elle kurulum zahmetli bir işlemdir. Bu işi kolaylaştıran kurulum programıdır. Kurulum halinin doğru çalıştığını test ederek dağıtımınızı yapabilirsiniz. Ayrıca kurulum programları dosyaları sıkıştırdıkları için kur dosyası az yer kaplar. Kurulum sırasında kullanıcıya bilgi veren ses ve grafik görüntüler vardır. İlerleme durumunu gösteren çubuk sayesinde ne kadar zaman kaldığını anlayabiliriz. Diske dosyaların kopyalanması hızlı ve pürüzsüz olarak halledilir. Bu da kullanıcıya programınızın profesyonel olduğunu gösterir.



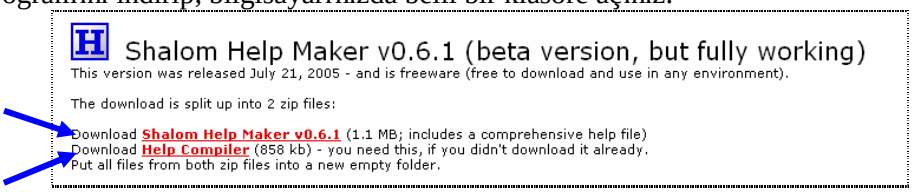
Resim 3.3: Bir programın kurulum ekranı

? Engelli insanlara bilgisayar kullanımında nasıl bir yardım edilebilir? Mesela gözleri görmeyen veya az gören insanlar nasıl bilgisayar kullanabilir?

Örnek 1:

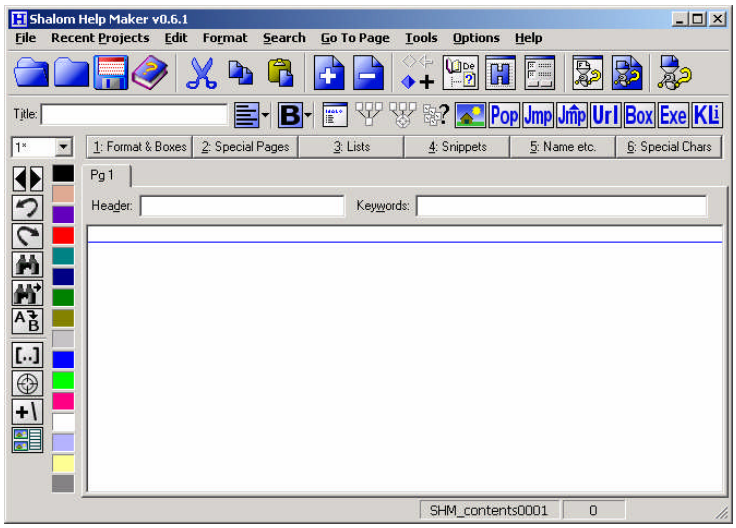
En kolay şekilde yardım dosyası hazırlayalım.

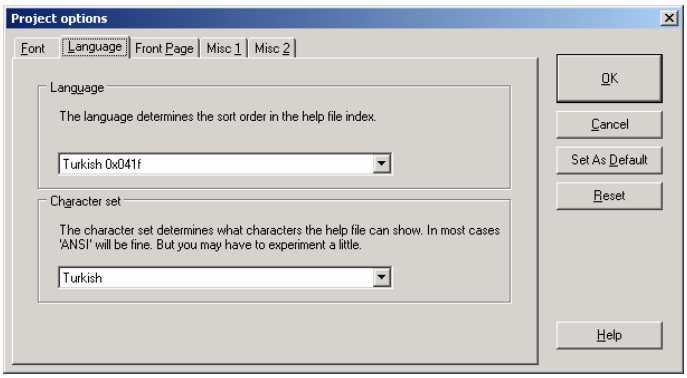
1. danish-shareware.dk/soft/shelpmbeta/index.html sitesinden "Shalom Help Maker" programını indirip, bilgisayarınızda belli bir klasöre açınız.



Resim 3.4: Her iki dosyayı indiriniz, dosyaları aynı klasöre açınız

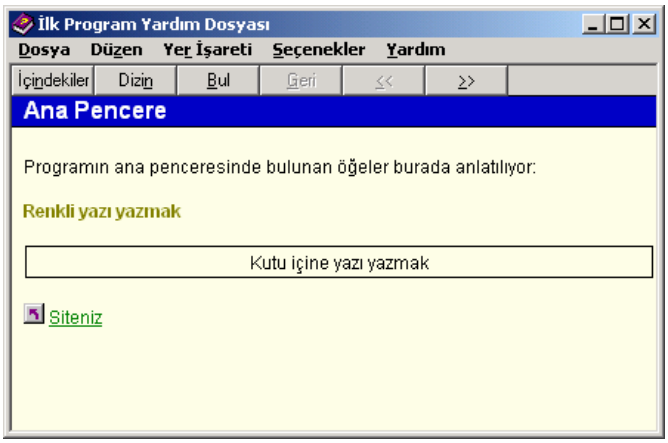
2. Ana program dosyası olan "shalomhelp.exe" dosyasını çalıştırınız.





Resim 3.5: Yardım dosyasının ilk hali ve F6 kısayolu ile açılan proje seçeneklerinden, Türkçe dil desteğinin sağlanması

3. Tasarım penceresinde sayfa sayfa dokümanlar yazılır, çok uzun olmayacak şekilde programın kullanımı anlatılır. Yeni sayfa eklemek için “Ctrl+N” kısayolu kullanılır. Her sayfanın anahtar kelimeleri “Keywords” kısmına noktalı virgüllerle ayrılarak yazılır.
4. Tasarımı denemek için F9 kısa yolu kullanılır. Bu kısa yol ile yardım dosyası derlenir ve çalıştırılır.



Resim 3.6: Yardım dosyamızın görünümü

Örnek 2:

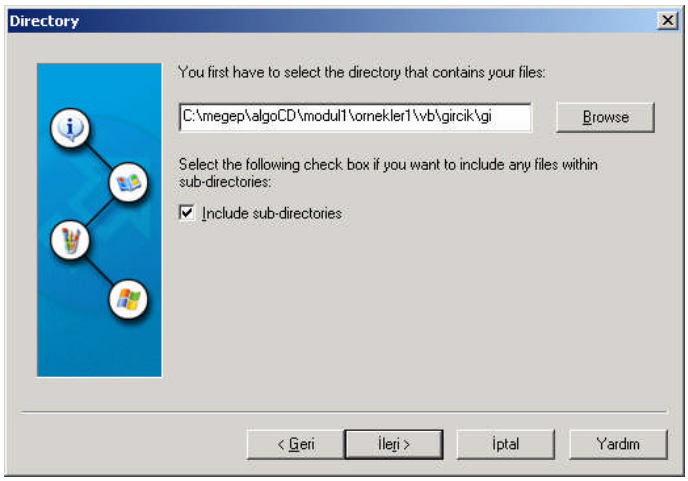
En kolay şekilde kurulum dosyası hazırlayalım.

1. www.clickteam.com/English/download_main.php?PID=4 sitesinden "Install Maker" programı indirip, bilgisayara kurulur.

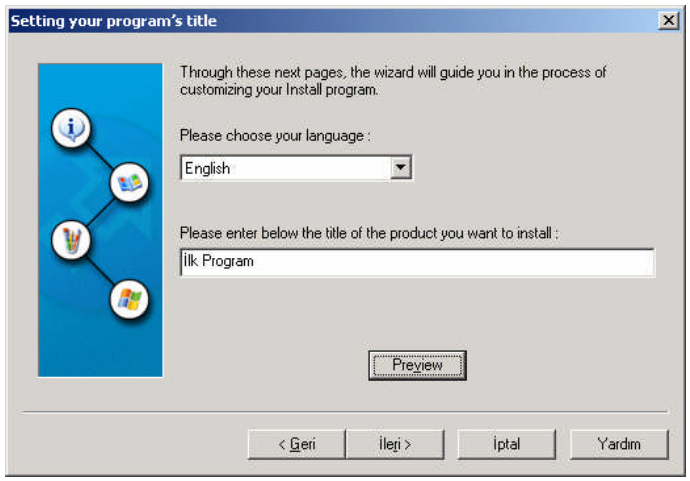
Resim 3.7: Kurulum hazırlama programı

2. Program açıldığında karşımıza gelen sihirbazı takip ederek, kurulumu hazırlarız.
3. Kurulum dosyasına kaynak dosyaları eklemek zorunda değilsiniz. Bu sebeple, gerekli dosyaları başka bir klasöre alabilir ve sihirbazda bu klasörü belirtebilirsiniz.
4. Oluşturulacak kısa yolları belirtiniz. Genellikle ana program ve yardım için kısa yol yapılır.
5. Sihirbaz sonlandırıldığında, kurulumu test etmek için, oluşan dosyayı çalıştırırız.
6. Başlat menüsünde oluşan kısa yolları çalıştırarak deneyebiliriz.
7. Programınızın deneme kurulumunu "Kaldır – Uninstall" kısa yolu ile bilgisayarınızdan silebilirsiniz.

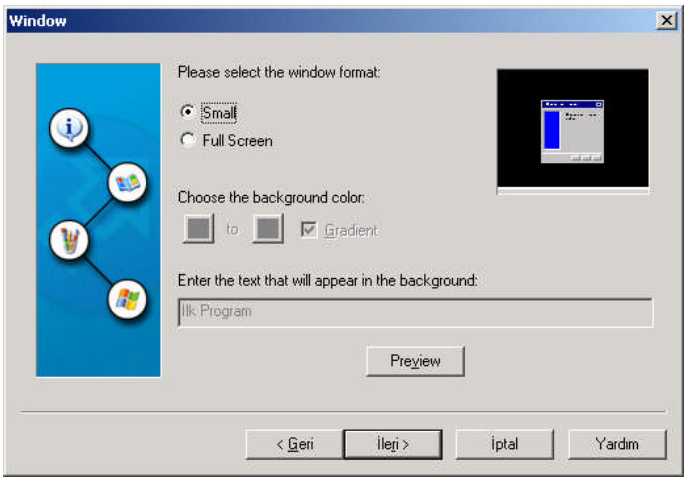
a)

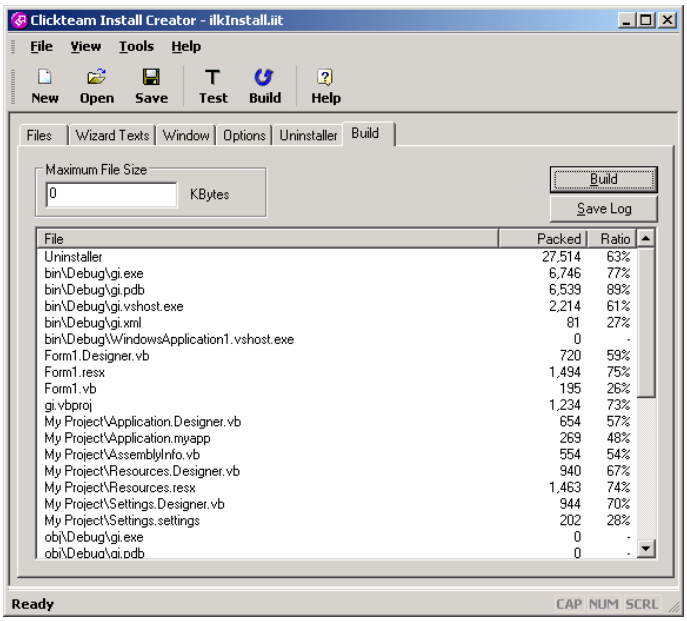

Directory
You first have to select the directory that contains your files:
C:\megep\algoCD\modul1\ornekler1\vb\gircik\gi
Browse
Select the following check box if you want to include any files within sub-directories:
Include sub-directories
< Geri
İleri >
İptal
Yardım

b)


Setting your program's title
Through these next pages, the wizard will guide you in the process of customizing your Install program.
Please choose your language :
English
Please enter below the title of the product you want to install :
İlk Program
Preview
< Geri
İleri >
İptal
Yardım

c)


Window
Please select the window format:
Small
Full Screen
Choose the background color:
to
Gradient
Enter the text that will appear in the background:
İlk Program
Preview
< Geri
İleri >
İptal
Yardım

ç) 

Resim 3.8: a, b ve c resimlerinde kurulum sihirbazı ve kurulum ayarlarının yapılması, ç resminde ise derleme işlemi (Ctrl+B) ve test işlemini (Ctrl+T) adımı vardır.

Aslında hiçbir programı bitiremezsiniz... Yalnızca üzerinde çalışmaya son verirsiniz.

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
1. İstediğiniz bir programın görsel ara birimini planlayınız.	Bilgisayarınızdaki yüklü programları inceleyiniz.
2. Bir görsel programlama dili ile kabaca programınızın prototipini tasarlayınız.	Tek form üzerine gerekli nesneleri mantıklı bir görünüm alacak şekilde yerleştiriniz. Takıldığınız yerlerde öğretmeninize danışınız.
3. Programınıza yardım dosyası ekleyiniz.	Yardım dosyası oluşturmak için gerekli programları araştırınız.
4. Programınızın kurulum dosyasını oluşturunuz.	Kurulum dosyası oluşturmak için gerekli programları araştırınız.
5. Yaptığınız prototipleri arkadaşlarınız ile paylaşınız.	Diğer bilgisayarlarda çalışabiliyor ise programınızın kurulumu bitmiştir.

ÖLÇME VE DEĞERLENDİRME

OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; sonunda parantez olanlar doğru / yanlış sorularıdır. Verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız.

1. Komutları düzenleyici ile yazar, derleyiciler ile makine diline çevririz. ( )

2. Programımızın yardım dosyası ile programcıya yardım bilgisi sağlarız. ( )

3. Web sayfaları derlenerek çalışırlar. ( )

4. Çoğu zaman iyi yazılan bir programı test etmeye gerek yoktur. ( )

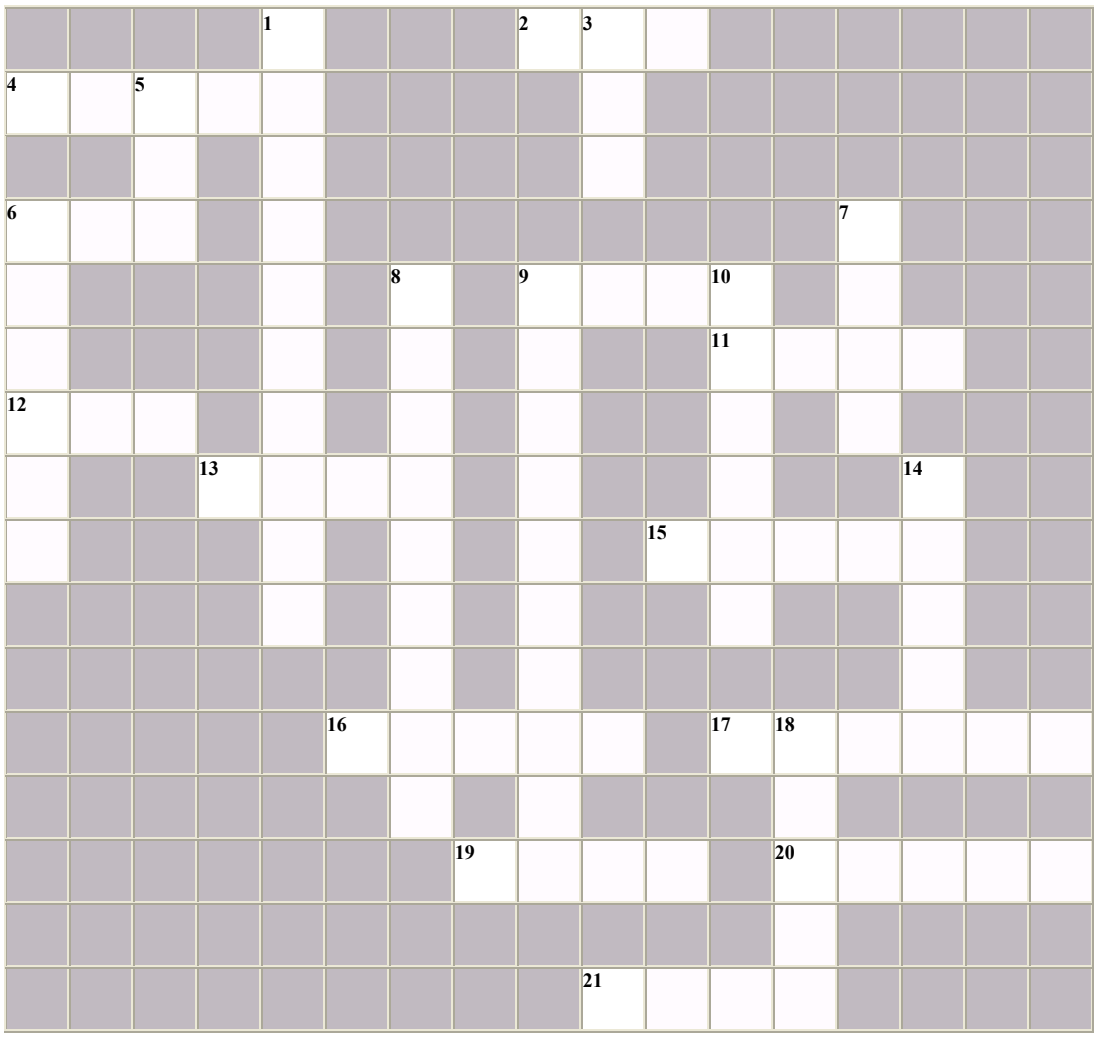
5. Hangisi bir programlama dilinin içinde bulunmak zorunda <u>değildir</u>?
A) Metin düzenleyici
B) Kurulum yapma programı
C) Böcek ayıklayıcı
D) Derleyici

6. Program yazmak için en kullanışlı program hangisidir?
A) Not defteri
B) Crimson Editor
C) Microsoft Word
D) Edit[‡‡‡]

7. Hangisi böcek ayıklama ile ilgili bir yöntem <u>değildir</u>?
A) Programı durdurmak
B) Adımlamak
C) Değişken izlemek
D) Durak noktaları yapmak

[‡‡‡] "Başlat*Çalıştır…*Edit" komutu ile çalıştırılabilir.

BULMACA



Aşağı

1. Bilgisayarın fiziksel kısımları
3. C dilinin sonraki sürümü
5. Yapısal sorgu dili
6. Görsel Pascal dili
7. Azaltılmış komut kümesi
8. PC'lere benzeyen diğer bilgisayar türü
9. HTML'ye yardımcı bir dil
10. Basit veritabanı programı
14. Microsoft firmasının veri tabanı yönetim programı
18. Delphi'nin Linux sürümü

Yan

2. Windows nesne kütüphanesi
4. Ücretsiz bir veri tabanı yönetim programı
6. Windows sistem kütüphanesi
9. C dilinden türetilen bir dil
11. Karmaşık komut kümesi
12. Programlanabilen mikro bilgisayar
13. Amerikan standartları kurumu
15. Windows çıkmadan önceki yaygın işletim sistemi
16. Temel metin dosya karakterleri
17. Bilgisayarın anladığı sayı sistemi
19. Web sayfalarının genel dili
20. Unix işletim sisteminin yaygın kullanılan diğer sürümü
21. C dili ile üretilmiş bir işletim sistemi

Değerlendirme:

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Doğru cevap sayınızı belirleyerek kendinizi değerlendiriniz. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt yaşadığınız sorularla ilgili konuları faaliyete geri dönerek tekrar inceleyiniz.

Modern olan şeyler diğerlerine göre hep göz alıcı olurlar.

MODÜL DEĞERLENDİRME

PERFORMANS TESTİ 1 (YETERLİK ÖLÇME)

Modül ile kazandığınız yeterliği, öğretmeniniz işlem basamaklarına göre 0 ile 8 puan arasında olacak şeklinde değerlendirecektir.

DEĞERLENDİRME KRİTERLERİ	**Puan**
Bilgisayarınızdaki programlama dillerini inceleme	
Temel bir programlama dili ve assembly komutlarını karşılaştırma	
Bilgisayarınızda yüklü olan bir görsel programlama dilini inceleme	
Web programlama dilleri ile ilgili araştırma yapma	
Akış diyagramını çıkarmak için gerçek hayattan bir problemi tespit etme	
Problem ile ilgili verileri belirleme	
Problem ile ilgili algoritma adımlarını belirleme	
Problem ile ilgili yaptığınız algoritmayı test etme	
İstediğiniz bir programın görsel ara birimini düşünme	
Bir görsel programlama dili kullanarak ile programınızı tasarlama	
Programınıza yardım dosyası ekleme	
Programınızın kurulum dosyasını oluşturma	
Toplam (en fazla 96 puan olabilir)	

ÖLÇME SORULARI

1. Bilgisayara ne yapacağının söylendiği komut topluluğuna ne denir?
A) Algoritma B) Program
C) Akış Şeması D) Olay şeması

2. Bir problemi sahte kod halinde yazarken, karar verilecek en son adım hangisidir?
A) Giriş B) Çıkış
C) İşlem D) Kullanıcı

3. Programın geliştirilme aşamasında, problemin tanımlandığı adım hangisidir?
A) Analiz B) Tasarım
C) Kod D) Test

4. Program geliştirmenin, tasarım aşaması ile _____.
A) Problem tanımlanır.
B) Çözüm planlanır.
C) Nesneler seçilir.
D) Algoritma programlama diline çevrilir.

5. Adım adım mantıksal sıralama hangisidir?
A) Akış Şeması B) Ara birim
C) Algoritma D) Böcek ayıklama

6. Programdaki hataların bulunması ve giderilmesi aşaması hangisidir?
A) Analiz B) Tasarım
C) Yardım dosyası ekleme D) Test ve böcek ayıklama

7. Hangisi algoritmanın programlama diline çevrilmesidir?
A) Analiz B) Tasarım
C) Kodlama D) Test ve böcek ayıklama

8. Programı tarif eden tüm materyallere ne denir?
A) Analiz B) Tasarım
C) Yardım dosyası D) Test ve böcek ayıklama

9. Hangisi bir görevin mantıksal sırasını grafiksel olarak resmeder?
A) Akış Şeması B) Sahte Kod
C) Olay Grafiği D) Nesne grafiği

10. Belli bir problemin taslak olarak Türkçe komut haline getirilmesine ne denir?
A) Akış Şeması B) Sahte Kod
C) Olay Şeması D) Nesne grafiği

11. Sembolleri birbirine bağlayan simge hangisidir?
A) akış çizgisi B) sonlandırma
C) yönler D) şartlar

12. _____ sembolü ile matematiksel işlemler yapılır.
A) Akış Çizgisi B) Bitiş
C) Karar D) İşlem

13. _____ sembolü ile program başlangıcı gösterilir.
A) Akış çizgisi B) Başlangıç
C) Karar D) İşlem

14. _____ sembolü ile semboller birleştirilir ve mantığın yönü gösterilir.
A) Akış çizgisi B) Başlangıç
C) Karar D) İşlem

15. _____ sembolü farklı akış çizgilerini birleştirir.
A) Akış çizgisi B) Başlangıç
C) Karar D) Bağlantı noktası

16. Hangisi sahte kodun dezavantajıdır?
A) Basittir.
B) Akış şemasından daha az yer kaplar.
C) Programlama diline benzer yazımı vardır.
D) Bir program kolay bir şekilde anlaşılabilir.

DEĞERLENDİRME

Yaptığınız değerlendirme sonucunda eksikleriniz varsa öğrenme faaliyetlerini tekrarlayınız.

Modülü tamamladınız, tebrik ederiz. Öğretmeniniz size çeşitli ölçme araçları uygulayacaktır. Öğretmeninizle iletişime geçiniz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1 CEVAP ANAHTARI

1	Y
2	Y
3	Y
4	Y
5	Y
6	D
7	C
8	B
9	C
10	C
11	C
12	D
13	B
14	D
15	C

ÖĞRENME FAALİYETİ-2 CEVAP ANAHTARI

1	D
2	Y
3	Y
4	Y
5	D
6	D
7	D
8	C
9	D

ÖĞRENME FAALİYETİ 3 CEVAP ANAHTARI

1	D
2	Y
3	Y
4	Y
5	B
6	B
7	A

BULMACANIN CEVABI

				1 E				2 O	3 C	X						
4 M	Y	5 S	Q	L					+							
		Q		E					+							
6 D	L	L		K									7 R			
E				T		8 M		9 J	A	V	10 A		I			
L				R		A		A			11 C	I	S	C		
12 P	I	C		O		C		V			C		C			
H			13 A	N	S	I		A			E			14 M		
I				I		N		S		15 M	S	D	O	S		
				K		T		C			S			S		
						O		R						Q		
					16 A	S	C	I	I		17 İ	18 K	İ	L	İ	K
						H		P				Y				
							19 H	T	M	L		20 L	I	N	U	X
												I				
									21 U	N	I	X				

MODÜL DEĞERLENDİRME CEVAP ANAHTARI

1	B
2	C
3	A
4	D
5	C
6	D
7	C
8	C
9	A
10	B
11	A
12	D
13	B
14	A
15	D
16	D

Cevaplarınızı cevap anahtarları ile karşılaştırarak kendinizi değerlendiriniz.

SÖZLÜK

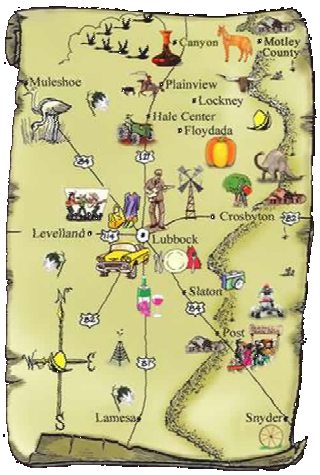
İsim	Okunuş	Anlam
access	ekses	erişmek
administrator	ıdmınıstreytır	idareci, süper kullanıcı
agent	eycınt	ajan, memur
application	eplikeyşın	uygulama
area	eria	alan
argument	argyumınt	parametre, argüman, fonksiyona veri gönderme
assemble	ıssembl	toplamak, bir araya getirmek
associate	essoşieyt	ilişkilendirmek
attribute	ettribyut	öznitelik, özellik
backup	bekap	yedekleme
base	beyz	taban, merkez, baz
batch	beç	toplu dosya
binary	baynıri	ikilik sayı sistemi
boolean	buuliın	mantıksal değer
boot	buut	bilgisayarın açılışı (bootup)
branch	branç	programın başka yöne dallanması
buffer	bafır	ön bellek, tampon, ara hafıza
bug	bag	böcek, hata
C++	si plas plas	C dilinin üst versiyonu
cache	keyç	ön bellek, kaşe
caret	karıt	^ işareti, şapka
cascaded	keskeydid	basamaklı, azalan
CGI	si ci ay	internette veritabanı uygulamaları
chain	çeyn	zincir
char	çar	tek karakterlik değer (character – keriktır)
circuit	sörkit	devre
cluster	klastır	küme, her track – iz'deki sektör grubu
code	kod	kod, programda kullanılan komut kümesi
collapse	kıleps	göçmek, gizlemek
command	kımand	komut
compare	kımpeyr	karşılaştırmak, comparison – karşılaştırma
component	kımpounınt	nesne, bileşik
conditional	kındişınıl	şart, koşullu
conflict	kınflikt	çakışma
console	konsoul	ekran ve klayve arabirimi (CON)
convert	kınvört	çevirmek, biçim değiştirmek
current	körrınt	geçerli, şu anki
cursor	körsır	imleç, fare oku (mouse pointer)
database	deytabeyz	veritabanı

Yazılım Terimleri Sözlüğü

YENİ	Yazılım eski sürümlerle tamamen uyumsuz.
İLERİ TASARIM TEKNİKLERİ KULLANILDI	Yönetim ne yaptığımızı pek anlamadı.
ÇIĞIR AÇAN YAZILIM	Neredeyse çalışıyor.
KENDİNE ÖZGÜ	Neler yaptığını yalnızca biz biliyoruz.
SAHADA TEST EDİLDİ	Valla biz hiç test etmeye fırsat bulamadık.
HERKES KULLANABİLİR	Kesinlikle esnek bir yazılım değil.
İLERİYE DÖNÜK	Kabul edilebilir bir performans için gereken donanım henüz piyasaya çıkmadı.
DESTEK GEREKTİRMEZ	Bir sorun çıkarsa düzeltmek olanaksız.
KALİTE STANDARTLARINA UYGUN	Hatasız derleniyor.
GARANTİLİ	Çalışmazsa size bir kopya daha göndeririz.
EŞSİZ	Neredeyse rakibimizin ürünü kadar iyi.
BENZERSİZ PERFORMANS	Bu kategorideki hiçbir yazılım bu kadar yavaş çalışmıyor.
YILLARIN DENEYİMİ	Sonunda bir kısmını olsun çalıştırmayı başardık.

ÖNERİLEN KAYNAKLAR

- bups.bups.bilkent.edu.tr/ibcomp/course/course.html
- computer.howstuffworks.com
- dan.corlan.net/bench.html
- en.wikipedia.org/wiki/Comparison_of_programming_languages
- en.wikipedia.org/wiki/Computer_bug
- en.wikipedia.org/wiki/Prolog
- www.alicebot.org
- www.analystdeveloper.com
- www.bilgisayarogren.com
- www.brianhetrick.com/casio/tpre.html
- www.btturk.net
- www.ceebot.com
- www.ceturk.com
- www.ceviz.net
- www.crimsoneditor.com
- www.delphiturk.com
- www.elk.itu.edu.tr/~caliskan/Harzemli.doc
- www.emu8086.com
- www.eng.iastate.edu/efmd/161algor.htm
- www.geocities.com/tablizer/langopts.htm
- www.google.com
- www.haber7.com/haber.php?haber_id=141185
- www.hazirkod.com
- www.icsharpcode.net
- www.ileriseviye.org
- www.math.uni-hamburg.de/it/software/fpk/ref/node1.html
- www.msakademik.net
- www.music.columbia.edu/cmix/algo.html
- www.nos.org/htm/basic2.htm
- www.osdata.com/topic/language/asm/asmintro.htm
- www.programmersheaven.com
- www.roesler-ac.de/wolfram/hello.htm
- www.scism.sbu.ac.uk/law
- www.sorucevap.com
- www.sourgeforge.net
- www.theadvisors.com/langcomparison.htm
- www.tpub.com/progintro
- www.tubider.org.tr/etik.html
- www.turing.org.uk/turing/scrapbook/computer.html
- www.vbturk.net
- www.yunus.projesi.com
- www2.latech.edu/~acm/HelloWorld.shtml

KAYNAKÇA

- Anadolu Üniversitesi Açıköğretim Sistemi (www.aof.edu.tr)
- AYFER Can Uğur, **Kim Güler Bilgisayarlara?**, Pusula Yayınevi, İstanbul, 1998
- AYDIN Serkan, **Programlama Ders Notları**
- BAĞRIYANIK Tarık, **Programlama Ders Notları ve Uygulamalı Genel Programlama Kitabı** (www.yunus.projesi.com)
- DURMAZ Alparslan, **Programlama Ders Notları**
- EKER Mustafa, **Algoritmayı Anlamak**, Nirvana Basımevi, Ankara, 2005
- NIIT Global Learning Solutions, Fundamentals of Programming
- MAGUIRE Steve, **Writing Solid Code**, Microsoft Basımevi, Washington, 1993
- TAN Ali, **Programlama Ders Notları**
- WALLACE Wang, **Beginning Programming for Dummies**, Wiley Basımevi, Indianapolis, 2004