7. VERİ YAPILARI (data stuctures) (14ncü hafta)

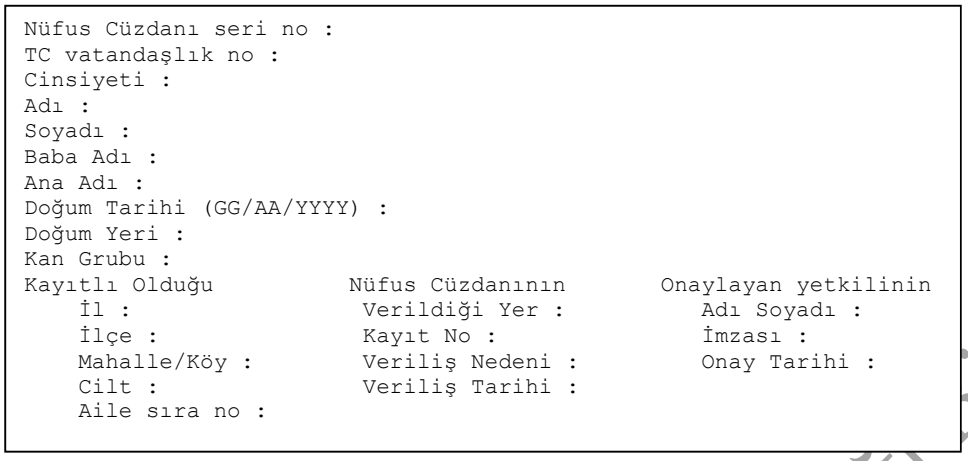
Veri, bir nesne için gerekli bilgilerin derlendiği topluluk, grub veya küme, sınıf şeklinde ifade edilebilir. Burada nesne ile ifade edilmek istenilen şey, bilgisayar ortamında değerlendirilecek bilgileri veya özellikleri olan bir cisim için uygun yazılımın geliştirilmesidir. Nesne (hedef) bir malzemenin üretimi ile bilgileri içerebilir. Bu ürünü tanımlayabilmek için gerekli bilgiler: ürünün adı, üretim tarihi, üretim seri nosu, rengi, boyutları, kaçıncı parça olduğu, fiyatı, katma değer vergisi ve benzeri bilgileri içerebilir. Nesne (bilgisayar yazılımı hazırlanacak) bir vatandaşa ait bilgiler olabilir: vatandaşın adı, soyadı, annesinin adı, babasının adı, doğum yeri, doğum yılı, mahallesi, sokağı vs. Bir işyerindeki çalışan bir nesne olabilir : Çalışanın adı, soyadı, sicil numarası, doğum tarihi, işe başladığı tarih, izinleri, evli olup olmadığı vs. Nesne tatil için bir otelde yer ayırtma işlemi olabilir : tatilcinin adı soyadı, otele giriş çıkış tarihleri, ücreti, konaklama ücretini ödeme şekli, cep telefonu, adresi vs. Nesne bir otobüs/uçak firmasından alınan bilet olabilir : hareket günü, saati, hareket yeri gibi bilgiler (veri). Kısaca (kendisi için program yazılacak) nesneye ait bilgiler veridir. Bu verilerin uygun bir şekilde saklanması ve kullanılması gerekmektedir. Veri yapısı, farklı uzunluklardaki çeşitli bilgilerden (veya tiplerden) oluşturulmuş bir veri grubunun yekpare olarak nasıl saklanacağının tanımlanmasıdır. Şekil 7-1 de bir personele ait Sicil No, Adı, Soyadı, Departmanı, Görevi gibi bilgiler içeren kart gösterilmektedir. Kart üzerindeki bilgiler veri tabanının nasıl olacağınıda belirler.

Şekil 7-1 Şiket personeline ait bilgiler.

Burada nesne şirket çalışanıdır. Sınıf olarak tanımlayabileceğimiz bu bilgi bu sınıfa girecek kişilerin bilgilerinin aynı olabileceğinide gösterir. Sınıf bilgileri toplu olarak okunabilir ve yazdırılabilir. Bu arkadaşlarınıza ait bilgileri topladığınız bir defter gibi düşünebilirsiniz. Defterin her sayfasına bir arkadaşınıza ait gerekli bilgileri yazabilirsiniz. Bu sayfaların her biri kayıt (record) olacaktır. Arkadaşlarınıza ait bilgiler topluluğu ise (programlama dilinin veri tabanına göre) nesnedir.

Bir nufüs cüzdanındaki bilgiler sınıf (veri yapısını) oluşturur. Verinin sınıfı (burada vatandaşlık bilgileri) aşağıdaki Şekil 7-2 gibi olabilir. Nüfus bilgileri bir veri tabanının şeklini oluşturur. Bu şablona göre veriler kayıt edilir, saklanır veya kullanılır.

C/C++ Ders Notları-A.Ü. Fizik Mühendisliği Bölümü



Şekil 7-2 Bir nüfus cüzdanı örneğindeki bilgiler veri yapısının şeklini belirler.

C/C++ programlama dilinde struct veri yapılarının genel yazım şekli aşağıdaki gibidir:

```
struct model_ismi {
  tip1 eleman1;
  tip2 eleman2;
  tip3 eleman3;
  .
  .
} nesne_ismi;
```

Buradaki `model_ismi` veri yapısının ismi için ve `nesne_ismi` ise program içinde kullanılacak veri yapısının (yani değişkenler topluluğunun) ismidir. Nesne ismi veri yapısının tanımlanması esnasında isteğe bağlı olarak yazılabilir veya yazılmayabilir. Nesne ismi (program satırları arasında) geçerli bir tanımlayıcıdır (veya tanımlayıcılardır). Küme parantezleri `{ }` ile tipleri ve onun alt tanımlayıcıları ile bir yapının elemanlarının oluşturdukları belirtilir.

Veri yapısı tanımında `model_ismi`, veri yapısına uygun geçerli bir ismi olmalıdır. Örneğin:

```
struct Urunler {
  char isim[30];
  float fiyat;
} Ayakkabi, Canta, Tshort;
```

Burada veri yapısının model ismi `Urunler` olarak tanımlanmıştır. Daha sonra bu veri yapısında veri bilgilerini saklamak için herbiri farklı tipte olan `isim` ve `fiyat` veri yapısı değişkenleri olarak tanımlanmıştır. Örnekte veri tanımı aşamasında nesneler belirlenmiştir.

Deklarasyon aşamasında yazımı seçimlik olan `nesne_ismi` alanı daha sonraki program kısımlarına bırakılabilir. Yukarıdaki örnektede böyle yapılmıştır. Örneğin yapı tipindeki nesneler `Ayakkabi`, `Canta` ve `Tshort` veri yapısı türündedir. Daha sonra bu veri yapısı türünde (`Urunler`) üç nesne olarak deklere edilmiştir : `Ayakkabi`, `Canta` ve `Tshort`.

```
struct urunler {
  char isim[30];
  float fiyat;
};
```

C/C++ Ders Notları-A.Ü. Fizik Mühendisliği Bölümü

```
urunler Ayakkabi, Canta, Tshort;
```

Burada veri yapısı **model** ini ve veri yapısı *nesnesi* arasındaki farkı kavramak veya belirlemek çok önemlidir. Değişkenler, `model` in tipini, ve `nesne` de değişkeni tanımlar. Tek bir `model` (tip) den birçok `nesne` (değişken) oluşturulabilir. Belirlenen yapı modeli için deklere edilmiş üç nesne (`Ayakkabi`, `Canta` ve `Tshort`) için oluşturulan farklı bellek alanları artık kullanılabilirdir. Veri yapısı değişkenlerinin kullanımı için nesne ve alan adları arasına nokta (`.`) sembolü yerleştirilir. Bu elemanlar (veya herhangi birisi) ile artık bunlar değişkenmiş gibi kullanılabilir:

```
Ayakkabi.isim – Ayakkabı adı ile ilgili bilgi girilebilir
Ayakkabi.fiyat – Ayakkabının fiyatı ile ilgili bilgi girilebilir
Canta.isim
Canta.fiyat
Tshort.isim
Tshort.fiyat
```

Bunların herbiri ile şu veri tipleri ortaya çıkar: `Ayakkabi.isim`, `Canta.isim` ve `Tshort.isim` elemanları `char[30]` tipinde, ve `Ayakkabi.fiyat`, `Canta.fiyat` ve Tshort.`fiyat` değişkenleri ise tek duyarlı değişken tanımlayıcı olan `float` tipindedir.

Örnek 7-1 Filmlerin afiş başlığı ve yapım yılı bilgileri nesne olarak kabul edilmektedir. Bu bilgileri ekrana yazan bir C++ programı aşağıda verilmektedir. Film başlığı 50 karakterlik bir dizide saklanmakta ve yapım yılı tam sayı olarak tanımlanmıştır.

Program 7-1 Filmlerin adının ve yapım yılının veri yapısı ile saklanması.

```
#include <iostream.h>
#include <string.h>
#include <stdlib.h>
#include <conio.h>
struct filmler { //veri yapısı tanımlanır
  char baslik[50]; // film başlığı için 50 elemanlı karakter dizisi
  int yil; // filmin üretildiği yıl için tam sayı değişkeni
} macera, belgesel; // program içinde kullanılcak nesneler tanımlanır
void filmyaz(filmler film); // fonksiyon prototipi tanımı
int main () // ana fonksiyon başlangıcı
{ strcpy (macera.baslik, "2001 A Space Odyssey");// film başlığı
  macera.yil=1968; // yıl bilgisi veri yapısı değişkenine değer aktarılır
  cout << "Macera filmi:\n ";
  filmyaz (macera);
  strcpy(belgesel.baslik,"Büyük Kediler");
  belgesel.yil = 2003;
  cout << "Belgesel filmi:\n ";
  filmyaz (belgesel);
  getch();
```

C/C++ Ders Notları-A.Ü. Fizik Mühendisliği Bölümü

```
  return 0; // işletim sistemine 0 değeri yollanarak program sonlanır
}
void filmyaz (filmler film) // fonksiyon
{ cout << film.baslik;
cout << "(" << film.yil << ")\n";}
```

Örnek 7-2 Aşağıda çizelgedeki verileri "araba.dat" dosyasından C dilinin yapı özelliğini kullanarak okuyan bir programın parçası verilmektedir. Bir arabaya ait şasi numarası, markası, modeli, üretim yılı, araç sahibinin adı soyadı bilgileri bir veri tabanın nasıl oluşturulacağını belirler.

Şasi seri no	Marka	Model	Renk	Üretim yılı	Adı soyadı
01289	Fiat	Doğan	Kırmızı	1989	Ali Dağ
14325	Mercedes	D200	Siyah	1985	Selim Dal
43124	Fiat	Marea	Mavi	1999	Ülkü Çan
67312	Ford	Focus	Gri	1997	Ayşe Anıl
09856	Ford	Mondeo	Yeşil	1996	Ünal Keçe
90715	Opel	Astra	Krem	1993	Hasan Kasa

Program 7-2 Arabalara ait bilgilerin veri tabanı.

```
#include <iostream.h>
#include <fstream.h>
#include <string.h>
#include <stdlib.h>
#include <conio.h>
struct oto { // veri yapısı tanımlanır
  char sasi[5];  // şasi nosu değişkeni
  char marka[10]; // markası değişkeni
  char model[10]; // model değişkeni
  char renk[10]; // renk değişkeni
  char uretim[10]; // üretim yılı değişkeni
  char sahibi[20]; // sahibi değişkeni
} araba; // nesne tanımlanır
int main(){char DosyAdi[11];
ifstream dosya1;  // dosya1 yazma amaçlı tanımlanır
cout << "Verilerin bulunduğu dosya adını giriniz : ";
   cin >> DosyAdi;
dosya1.open (DosyAdi, ios::in);
cout << "Girilen dosya adı : " << DosyAdi << endl << endl;
//while( ! feof(dosya1))
while(dosya1) {
dosya1.get(araba.sasi, 5);
dosya1 >> araba.sasi;
```

C/C++ Ders Notları-A.Ü. Fizik Mühendisliği Bölümü

```
dosya1.get(araba.marka, 10);
dosya1 >> araba.marka >> araba.model >> araba.renk ;
dosya1 >> araba.uretim >> araba.sahibi;
cout << "Arabanın Şasi No :" << araba.sasi << endl;
cout << "Arabanın markası :" << araba.marka << endl;
cout << "Arabanın modeli  :" << araba.model << endl;
cout << "Arabanın rengi   :" << araba.renk << endl;
cout << "Arabanın üretimi :" << araba.uretim << endl;
cout << "Arabanın sahibi  :" << araba.sahibi << endl;
cout << "---------------------------------" << endl;
}
getch(); return 0; }
```

Örnek 7-3 Hava durumuna ait bilgileri saklamak için veri tabanına yazılacak bilgiler (C dilinin yapı özelliğini kullanarak) için bir program parçası aşağıda verilmektedir. Saklanacak bilgiler istasyon adını, ölçüm tarihi, sıcaklığı, nem oranını, rüzgar hızını içermektedir.

Program 7-3 Hava durumu bilgileri (veritabanı).

```
#include <iostream.h>
#include <string.h>
#include <stdlib.h>
#include <conio.h>
struct hava { // veri yapısı tanımlanır
  char IstasyonAdi[10];  // hava istasyonunun adı değişkeni
  char Tarih[10]; // tarih bilgisi değişkeni
  float Sicaklik; // sıcaklık değişkeni
  float NemOrani; // nem oranı değişkeni
  int RuzgarHizi; // rüzgarın hızı değişkeni
} hava; // program içinde kullanılacak nesne tanımlanır
```

7.1 Yapısal Elemanlara Değer Atanması

Yapının elemanlarının tiplerinin mutlaka bilirtilmesi gerekir. Veri yapısında tanımlanan elemanlar derleyici tarafından bilinmez, derleyici yapının nesne ismini bilir. Veri yapısı önceden tanımlanarak programın veri için ne kadar bilgi saklama alanına ihtiyaç duyduğu ortaya konmuş olur. Böylelikle yapının alt alanları tanımlanmış olur.

Örnek 7-4 Tarihleri bilgisayar belleğinde saklamak için hazırlanması düşünülen bir veri yapısı aşağıda gibi olabilir:

```
struct kayit
{ int  gun;
  int  ay;
  int  yil;
} tarih;
```

Buradaki tarih yapısının elemanlarına bilgilerin veya değerlerin atamaları aşağıdaki gibidir:

C/C++ Ders Notları-A.Ü. Fizik Mühendisliği Bölümü

```
tarih.gun = 21;
tarih.ay = 07;
tarih.yil = 1985;
```

Program 7-4 Tarihleri veri tabanı yapısı kullanarak saklamak için hazırlanması düşünülen bir C programı aşağıda verilmektedir.

```
#include <stdio.h>
struct kayit {/* tarih yapısı tanımlanır */
  int gun;
  int ay;
  int yil;};
int main(){
  struct kayit tarih;
  tarih.gun = 10;
  tarih.ay = 12;
  tarih.yil = 1995;
  printf("Bugünün tarihi %d/%d/%d.\n",tarih.gun, tarih.ay, tarih.yil );}
```

Örnek 7-5 Bağış yapan kişilerin adlarını ve soyadlarını yapı elemalarında saklayan bir C programı aşağıda verilmektedir. Bağışçı kişilerin ad ve soyadı için 30 elemanlı karakter tipindeki diziler tanımlanmıştır. Bağış miktarları ise tek duyarlı sayı olarak tanımlanmıştır.

Program 7-5 Bağışçıların bilgilerinin saklanması.

```
#include <stdio.h>
/* yapının tanımlanması ve veri saklanması. */
/* tek duyarlı değişken ve iki tane karakter tipinde dizi. */
struct data{
     float miktar;
     char adi[30];
     char soy[30];
} kayit;
 main(){
   printf("Adı ve soyadı aralarında,\n");
   printf("boşluk olacak şekilde giriniz: ");
   scanf("%s %s", kayit.adi, kayit.soy); /* klavyeden giriş. */
   printf("\nBağış miktarını giriniz : ");
     scanf("%f", &kayit.miktar);
     /* bilgiler ekranda görüntülenir. */
     /* Not: %.2f tek duyarlı sayının yazım formatı */
     /* virgülden sonra iki hane görüntülenir */
     printf("\n %s %s  %.2f TL bağış yapmıştır.\n", kayit.adi, kayit.soy,
             kayit.miktar);
     return 0; }
```

C/C++ Ders Notları-A.Ü. Fizik Mühendisliği Bölümü

Programın çalıştırılması sonucunda elde edilen ekran çıktısı aşağıdaki gibidir:

```
Adı ve soyadı aralarında,
boşluk olacak şekilde giriniz: Ali Genç
Eağış miktarını giriniz : 1000.00
Ali Genç 1000.00 TL bağış yapmıştır.
```

7.2 İşaretçiler-Poınters

İşaretçiler karmaşık veri yapılarının kolayca kulanılmasını sağlar, arguman değerlerinin değiştirilerek fonksiyona aktarılması, dinamik bellekle çalışılması açısından ve dizilerle ilgili işlemlerde çok kullanışlı olmaktadırlar. İşaretçi ile bir veri başlığına dolaylı yollardan ulaşılır. Aşağıda işaretçiler için basit örnekler verilmektedir:

```
int  count = 10, *sayac;
```

`count` tamsayı değişkenine 10 değeri atanır, `sayac` isimli pointer tanımlanır. Değişkenin önündeki * işareti değişkenin pointer olduğunu göstermektedir. `sayac` ve `count` arasındaki doğrudan olmayan ilişki `&` işareti ile tanımlanır. Aşağıdaki aktarma işleminde `count` değişkenindeki değer `sayac` işaretcisine (tanımladığı bellek bölgesine) aktarılır:

```
sayac = &count
```

İşaretçiler değer içermezler, bellek adresleri içerirler!

Örnek 7-6 Aşağıda Normal bir değişkene işaretcideki değerin atanması veya işaretcideki değerin normal bir değişkene aktarılmasını gösterilmektedir..

```
#include <stdio.h>
#include <conio.h>
main(){
int count  = 10, x, *sayac;
sayac = &count; /* count un bellekte sakladığı değer sayac a aktarılır */
x = *sayac; /* sayac in bellek adresindeki değeri x e aktarılır */
printf("count = %d, x = %d\n", count, x);getch();
}
```

Örnek 7-7 Aşağıda işaretçilerin değişik bir şekilde karakterlerle birlikte kullanımı gösterilmektedir,

```
#include <stdio.h>
main()
{
char c = 'Q';
```

C/C++ Ders Notları-A.Ü. Fizik Mühendisliği Bölümü

```
char *char_pointer = &c;
printf("%c %c\n", c, *char_pointer);
c = 'Z';
printf("%c %c\n", c, *char_pointer);
*char_pointer = 'Y';
/* char_pointer ın bellekteki adresi Y olur */
printf("%c %c\n", c, *char_pointer);
}
```

Örnek 7-8 Kişilerin adları ve onlara verilen numaraları pointer olarak tanımlanmış yapı elemalarında saklayan bir C programı aşağıda verilmektedir.

Program 7-6 Kişi bilgilerinin veri yapısı ile saklanması.

```
/* yapılardaki dizilerin pointer larla birlikte kullanılaması */
#include <stdio.h>
#define MAX 4
/* önce yapı tanımlanır, deklare edilir ve sonra ilk değer */
/* atamaları yapılır. */
struct part {
    int numara;
    char ad[10];
} data[MAX] = {1, "Veli",
               2, "Ahmet",
               3, "Ali",
               4, "Muharrem"
               };
/* part yapısında p_part isimli pointer ve sayac tanımlanır. */
struct part *p_part;
int sayac;
main(){
/* pointer a dizi elemanının ilk değeri aktarılır. */
   p_part = data;
/* dizi döngü içindedir, pointer artırılır */
   for (sayac = 0; sayac < MAX; sayac++){
      printf("%d adresindeki değer: %d %s \n ", p_part, p_part->numara,
           p_part->ad);
           p_part++;}
   return 0;}
```

Programın çalıştırılması sonucunda elde edilen ekran çıktısı aşağıdaki gibidir:

```
96 adresindeki değer: 1 Veli
```

C/C++ Ders Notları-A.Ü. Fizik Mühendisliği Bölümü

```
108 adresindeki değer: 2 ahmet
120 adresindeki değer: 3 Ali
132 adresindeki değer: 4 Muharrem
```

7.3 Sınıflar

Yukarıda açıklanan yapılara benzer bir yapı türüde verilerin sınıflarla tanımlanmasıdır. C++ da sınıf, veri ve fonksiyonları aynı yapı içinde organize etmek için kullanılan mantıksal yöntemdir. Sınıflar, C de sadece veri değişkenlerini eleman olarak kabul eden **`struct`** anahtar kelimesi ile tanımlanan yapıdan farklı olarak **`class`** anahtar kelimesi ile deklere edilirler ve fonksiyonlarıda değişken eleman olarak kabul ederler. Yani sınıfın (classes) yapıdan (struct) farkı içinde fonkiyon barındırabilmesidir. Sınıfların yazım şekli aşağıdaki gibidir:

```
class sinif_ismi {
  izinli_etiket_1:
    eleman1;
  izinli_etiket_2:
    eleman2;
  ...
  } nesne_ismi;
```

Yukarıdaki **`sinif_ismi`** sınıfın adıdır (kullanıcı tanımlı *tip* ve bu struct da model ismi olarak geçmiştir) ve **`nesne_ismi`** program içinde geçerli nesne tanımlayıcısı olmak üzere bir veya birkaç isim şeklinde tanımlanabilir. Deklerasyonun ana kısmı, veri veya fonksiyonları içerecek şekilde **`elemanlari`** içerir (model ismi ile veri yapısı tanımlanırken, nesne ismi ile bu veri grubunun program içindeki adıdır). **`izinli etiketler`** ise anahtar kelimeler olan **`private:`**, **`public:`** veya **`protected:`** şeklinde olabilir. Bu anahtar kelimelerle *elemanlar* aşağıdaki gibi referans edilebilir:

- **`private`** elemanlarına sınıfın diğer elemanları tarafından veya "*arkadaş*" sınıflar tarafından erişilebilir.
- **`protected`** elemanlarına benzer sınıf elemanları *arkadaş* sınıflar veya bunlardan türetilmiş sınıflar tarafından erişilebilirdir.
- **`public`** elemanlarına sınıfın görülebilir veya kullanılabilir olduğu heryerden ulaşılabilir.

Eğer bir sınıfın üyeleri herhangi bir izin etiketi olmadan tanımlanıyorsa bunlar **`private`** olarak algılanırlar, böyle sınıf üyeleri `class` anahtar kelimesi ile tanımlanabilir.

Örnek 7-9 Bir dörtgenin yüzey alanı ve çevresinin uzunluğunu hesaplamak için sınıf yapısı kullanılabilir. Burada nesne dörtgendir. Bu nesnenin elemanları ise kenar uzunlukları, yüzey alanı ve çevre uzunluğudur. Çevre uzunluğu ve yüzey alanı sınıf içinde fonksiyon olarak tanımlanmaktadır.

```
class Dortgen {
  private:
    int x, y;
  public:
    void deger_aktar (int,int);
    int alan (void);
    int cevre (void);
  } dortgen;
```

C/C++ Ders Notları-AÜ Fizik Mühendisliği Bölümü

Yukarıdaki sınıf tanımında, `Dortgen` isminde bir sınıf tanımlanmakta ve bu sınıfın nesnesi `dortgen` olarak isimlendirilmiştir (baş harflerin farklı olmasına dikkat ediniz). Bu sınıfın dört elemanı vardır: `private` kısmında iki değişken `int` (`x` ve `y`) olarak tanımlanmış ve `public` kısmında sadece protipleri olan iki fonksiyon tanımlanmıştır: `deger_aktar()`, `alan()` ve `cevre()`. Burada sınıf adı ile nesne adı arasındaki fark görülebilir. `Dortgen` sınıf adı (kullanıcı tanımlı tip) iken `dortgen Dortgen` tipinin nesnesidir. `int` class ve `int a` arasındaki farkı aşağıdaki gibi tanımlanabilir:

```
int a;
int sınıf_adı
```

Yeterli bilgiler ile, `dortgen` isimli sınıfın public elemanlarına normal fonksiyonlar veya değişkenler ile kolayca ulaşılabilir. Sınıf elemanlarına nesne adından sonra nokta (.) koyup sınıf üyesinin (değişkenin) adı yazılırarak ulaşılabilir. Örneğin :

```
dortgen.deger_aktar (3,4);
alan = dortgen.alan();
```

Örnek 7-10 Bir dörtgenin alanını ve çevresini sınıf yapısı ile hesaplayan program aşağıda verilmektedir. Dörtgenin kenarları veri yapısı içinde **private**, alan ve çevre bilgileri ise **public** olarak tanımlanmıştır.

Program 7-7 Dörtgen alanının ve çevresinin sınıf yapısı kullanılarak hesaplanması.

```
// sınıf kullanımına örnek
#include <iostream>
using namespace std;
class Dortgen{
private :
    int x, y;
  public:
    void deger_aktar(int a, int b)
    { x = a; y = b;}
    int alan () {return (x*y);}
    int cevre() {return (2*(x+y));}
};
int main () {
  Dortgen dortgen;
  dortgen.deger_aktar (3, 4);
  cout << "Alan : " << dortgen.alan();
  cout << "\n";
  cout << "Çevre: " << dortgen.cevre();
  cout << "\n";
  system("PAUSE");
```

C/C++ Ders Notları-A.Ü. Fizik Mühendisliği Bölümü

```
  return 0;}
```

Örnek 7-11 300 gram ağırlığındaki bir cismin x ekseni boyunca hareketi zamana bağlı olarak $x(t)=0.20t-5.0t^2+7.5t^3$ denklemi ile verilmektedir. Cismin hareketini konum, hız, ivme ve cisme etki eden kuvvet bilgilerini (sınıf olarak tanımlanan değişkenlerde) bir dosyaya yazan (saklayan) C++ programı aşağıda verilmektedir. Verileri t, x(t), $v(t)=dx(t)/dt=-10.0t+22.5t^2$ m/s, $a=d^2x(t)/dt^2=-10.0+45.0t$ m/s^2, F=ma bağımlı ve bağımsız değişklenlerine bağlı olarak bir dosyaya aşağıdaki şekilde yazdırılmaktadır.

```
Zaman(sn) konum(m) hız(m/s) ivme(m/s2) kuvvet(N)
 0.0       0.0      0.2      -10.0     -3.0
 0.5        .        .          .         .
  .         .        .          .         .
  .         .        .          .         .
```

Programda kullanılan değişkenler : t- zaman(sn), x-konum(m), v-hız(m/sn), a-ivme(m/s2), F-kuvvet(N), x(t)=0.20*t-5.0*t*t+7.5*t*t*t, v(t)=0.20-10.0*t+22.5*t*t, a(t)=-10.0+45.0*t, F=ma şeklindedir.

Program 7-8 x-ekseni boyunca zamana bağlı hareket : $x(t)=0.20t-5.0t^2+7.5t^3$

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <math.h>
using namespace std;
class f { public : float m, t, x, v, a, F;};
int main () {
  f hareket;
  ofstream dosya1;  // dosya1 yazma amaçlı tanımlanır
  dosya1.open ("veriler.txt", ios::out);
  hareket.m=0.300; // değişkenlere ilk değerleri aktarılır
  hareket.t=0.0;
  float dt=0.5;
  dosya1<<" Zaman(sn) konum(m) hız(m/s) ivme(m/s2) kuvvet(N)\n";
  while (hareket.t < 10.0)
  {
   hareket.x=0.20*hareket.t-
                  5.0*pow(hareket.t,2)+7.5*pow(hareket.t,3);
   hareket.v=0.20-10.0*hareket.t+22.5*pow(hareket.t,2);
   hareket.a=-10.0+45.0*hareket.t;
   hareket.F=hareket.m*hareket.a;
   dosya1 << fixed << setprecision(1);
   dosya1.width(4);
   dosya1.fill(' '); //belirtilen haneler boşlukla doldurulur
   dosya1 << hareket.t << "     ";
```

C/C++ Ders Notları-A.Ü. Fizik Mühendisliği Bölümü

```
   dosya1.width(4);
   dosya1 << setw(2);
   dosya1.fill(' '); //belirtilen haneler boşlukla doldurulur
   dosya1.width(4);
   dosya1 << hareket.x << "      ";
   dosya1 << setw(2);
   dosya1.fill(' '); //belirtilen haneler boşlukla doldurulur
   dosya1 << hareket.v << "      ";
   dosya1 << setw(2) << hareket.a << "     ";
   dosya1.fill(' '); //belirtilen haneler boşlukla doldurulur
   dosya1 << hareket.F << endl;
   // zaman değeri artırılır
   hareket.t=hareket.t+dt;
  }
  // dosya kapatılır
  dosya1.close ();  //1 dosya kapatılır
// program bir tuşa basana kadar bekletilir
  system("PAUSE");
  return 0;}
```

Örnek 7-12 Yarıçap değeri verilince bir dairenin alanını ve çevresini sınıf özelliğini kullanarak hesaplayan C++ programı aşağıda verilmektedir.

Program 7-9 Bir dairenin alanı ve çevresinin class kullanılarak hesaplanması.

```
#include <iostream>
#include <conio.h>
#include <math.h>
using namespace std;
class daire
{ private :
     float yaricap;
  public  :
     float DegerAktar (float yari)
     { yaricap=yari;}
  float alan()
     {return M_PI*pow(yaricap,2);}
  float dairecevresi()
     {return 2.0*M_PI*yaricap;}
} c1; // class  sonu
int main()
```

C/C++ Ders Notları-A.Ü. Fizik Mühendisliği Bölümü

```
{ c1.DegerAktar(3.5);
  cout << " Dairenin alan     = " << c1.alan() << endl;
  cout << " Dairenin ŧevresi = " << c1.dairecevresi() << endl;
  getch();
return 0;}
```

7.4 SORULAR

1. Rüzgarla elektrik üreten yel değirmenleri yüksek hava sürükleme kuvveti ile döner. Bir küre şeklindeki cisim için F_D sürükleme kuvveti r^2v^2 ile orantılıdır. Burada r yeldeğirmeninin dönen parçasının kol uzunluğu ve v rüzgarın hızıdır. Elde edilen güç $P=F_Dv=ar^2v^3$ ile orantılıdır. Denklemdeki a=2 $Watts^3/m^5$, r=1.5 m ve v=8 m/s lik durum için bu ev tipi yel değirmeninin ürettiği gücü r, v ve a yı sınıf değişkeni ve bu sınıfın içinde tanımlı olan güç değerini bir fonksiyon aracılığı ile hesaplayan bir C/C++ programını yazınız.

2. x-ekseni boyunca zıt yönlerde ilerleyen iki enine dalga atması aşağıdaki dalga fonksiyonlaı ile temsil edilebilir: $y_1(x,t)=6/(x-3t)^2$ ve $y_2(x,t)=3/(x+3t)^2$. Burada x ve y cm ve t saniye cinsindendir. y_1+y_2 bileşik dalga fonksiyonunun şeklini zamanın fonksiyonu olarak elde edebileceğiniz bir program yazınız. Zamanı 0ncı saniyeden 3.5 saniyeye kadar 0.5 er saniyelik aralıklarla artırınız. Yukarıdaki fonksiyonları ve değişkenlerini class çinde tanımlayınız.

3. Bir boyutlu eksen üzerinde zıt yönlerde hareket eden, çarpışmadan önceki hızları v_1, v_2, kütleleri m_1 ve m_2 olan iki cisim elastik olarak çarpışmaktadır (çarpışmadan önceki ve sonraki enerjiler korunmakta ve cisimler çarpışmadan sonra ayrışmaktadırlar). Cisimlerin çarpışmadan sonraki hızları $v_1'=(m_1-m_2)*v_1/(m_1+m_2)$ ve $v_2'=2m_1*v_1/(m_1+m_2)$ denklemleri ile verilmektedir. Cisimlerin çarpışmadan sonraki hızlarını class (sınıf) içinde fonksiyon kullanarak hesaplayan bir program yazınız. Bunun için çarpışmadan önceki, sonraki kütle ve hız değerlerini ayrı ayrı sınıflarda tanımlayınız. Cisimlere ait başlangıç verilerinin klavyeden girilmesini sağlayınız.

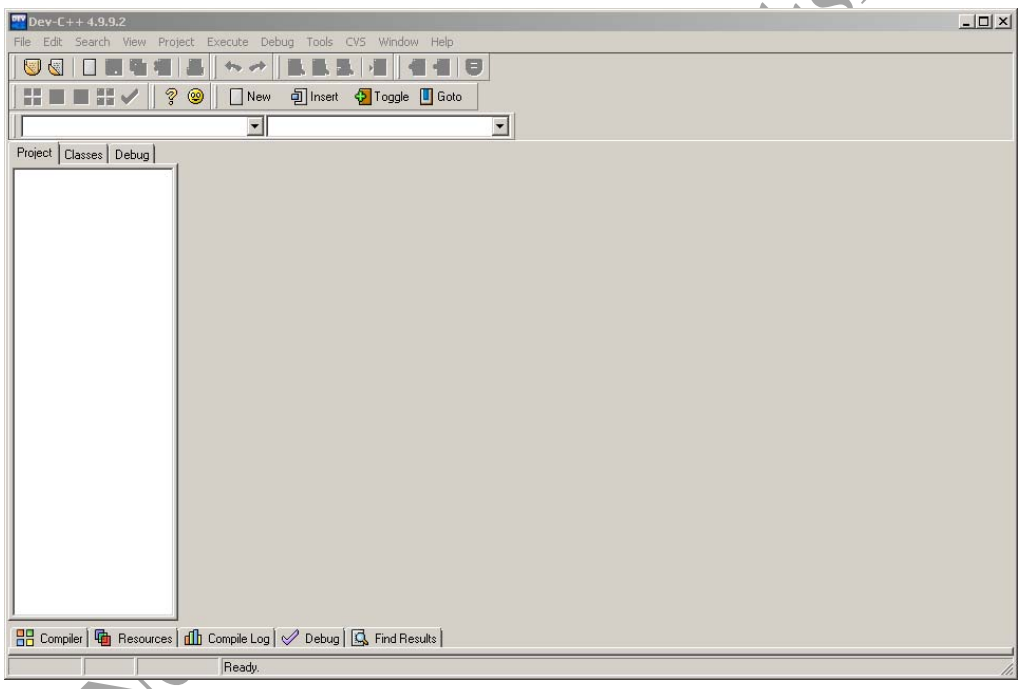
C/C++ Ders Notları-A.Ü. Fizik Mühendisliği Bölümü

8. EK 1. Dev-C++ nin kullanımı

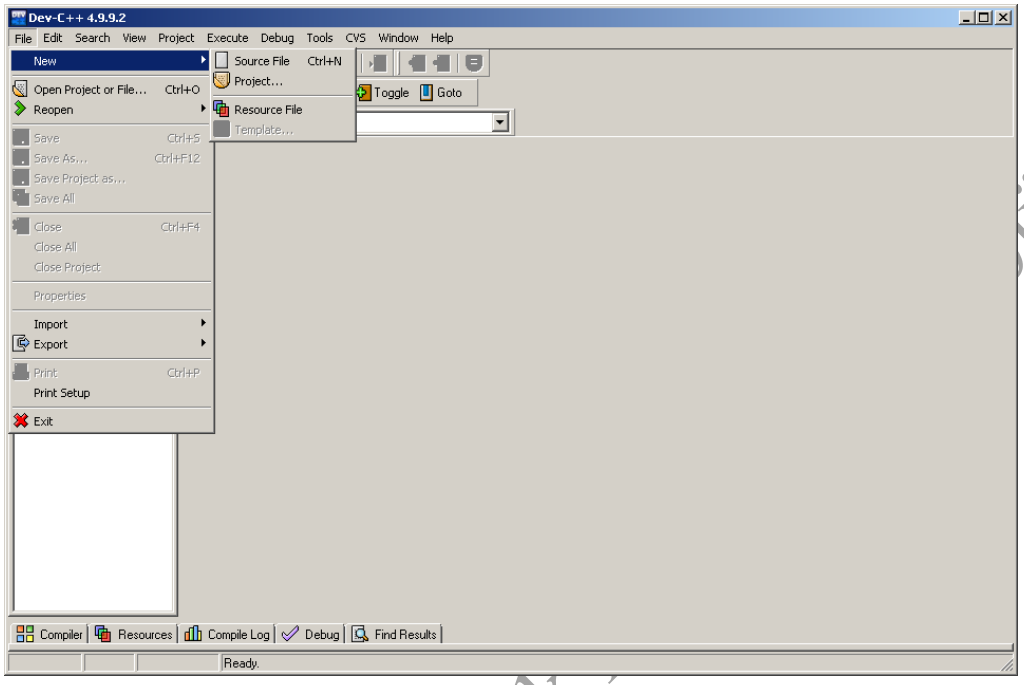
Adım 1: Yeni bir projenin oluşturulması (Create a new project).

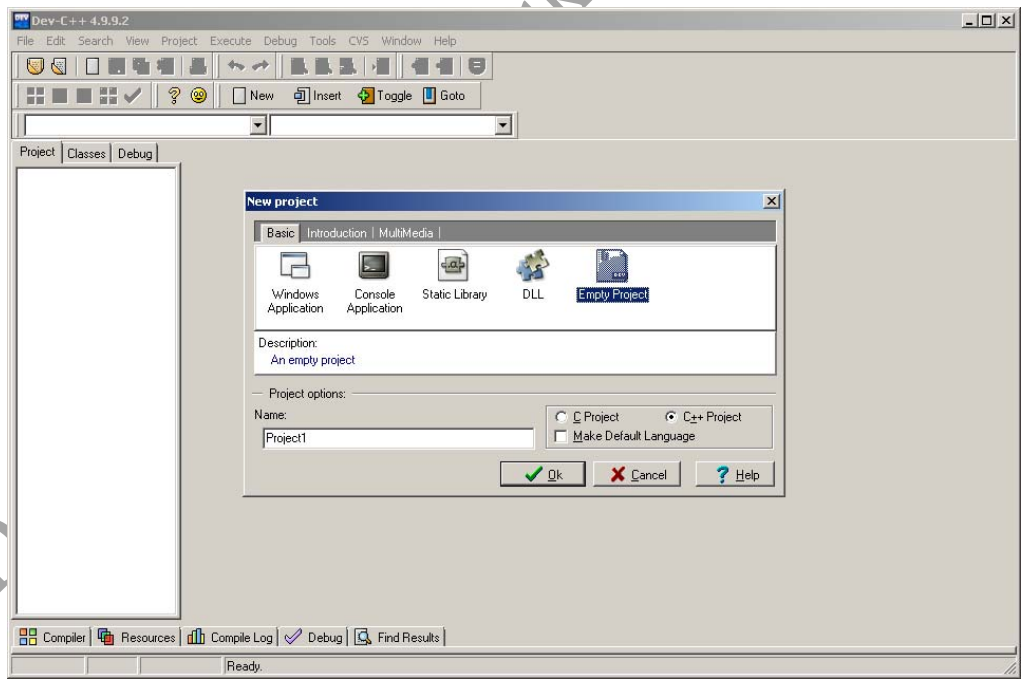
Bir "proje" bir programı derlemek için kullanılacak olan bütün elemanları bir araya toplar.

- "`File`" menüsünden "`New Project...`" (veya kısaca CTRL+N ye basınız).
- Buradan "`Empty Project`" i seçiniz, burada "`C++ project`" in seçildiğinden emin olunuz ve "`OK`" yi tıklayınız.
- Bu noktada Dev-C++ size projeniz için bir isim vermenizi isteyecektir. Bu aynı zamanda çalışacak programında adı olacaktır.

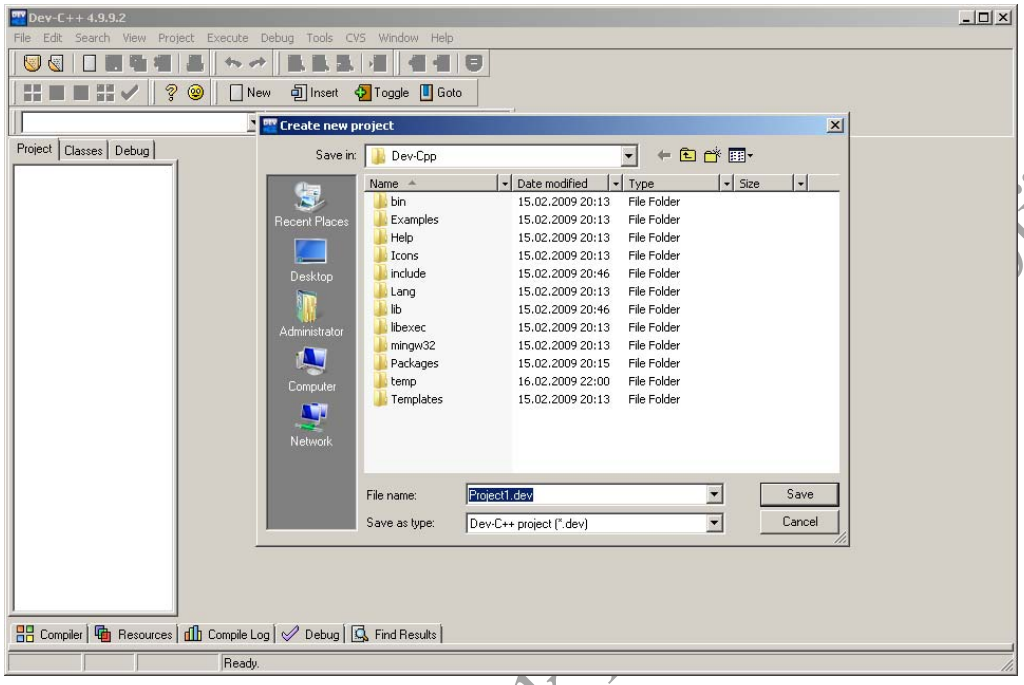


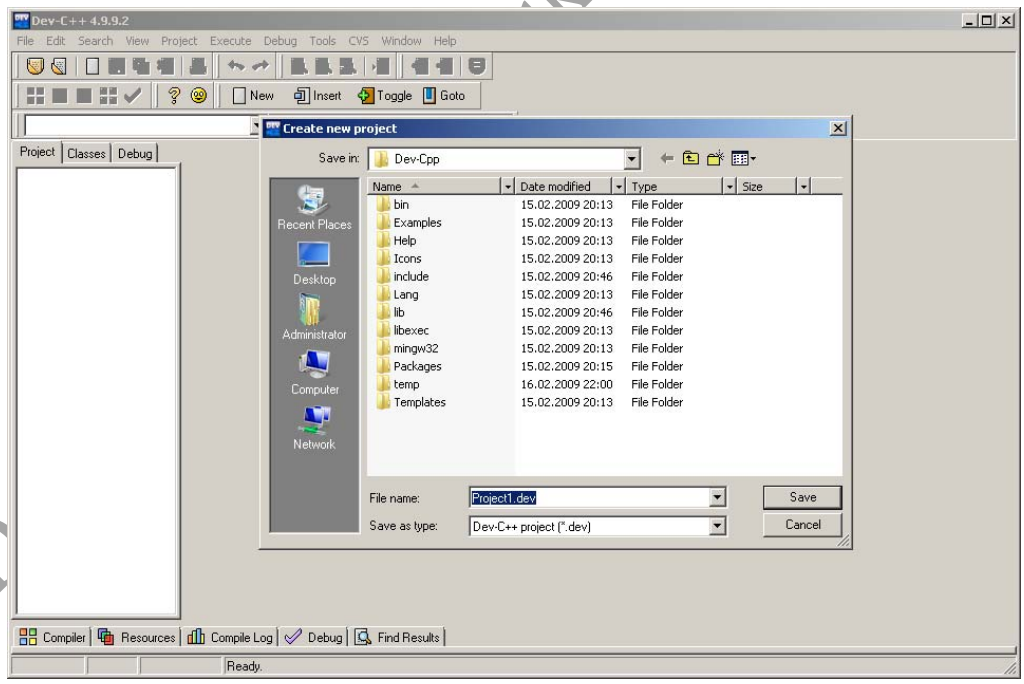
C/C++ Ders Notları Bilgisayar Mühendisliği Bölümü


Dev-C++ 4.9.9.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New
Source File Ctrl+N
Project...
Resource File
Template...
Open Project or File... Ctrl+O
Reopen
Save Ctrl+S
Save As... Ctrl+F12
Save Project as...
Save All
Close Ctrl+F4
Close All
Close Project
Properties
Import
Export
Print Ctrl+P
Print Setup
Exit
Toggle
Goto
Compiler
Resources
Compile Log
Debug
Find Results
Ready.

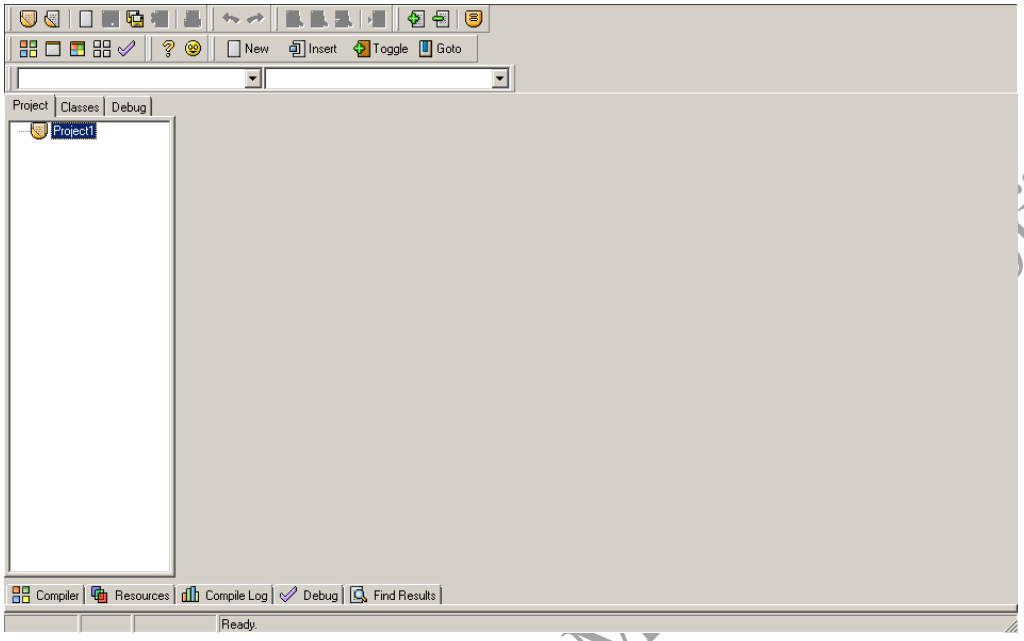

Dev-C++ 4.9.9.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New
Insert
Toggle
Goto
Project
Classes
Debug
New project
Basic
Introduction
MultiMedia
Windows Application
Console Application
Static Library
DLL
Empty Project
Description:
An empty project
Project options:
Name:
Project1
C Project
C++ Project
Make Default Language
Ok
Cancel
Help
Compiler
Resources
Compile Log
Debug
Find Results
Ready.

C/C++


Dev-C++ 4.9.9.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Classes Debug
Create new project
Save in: Dev-Cpp
Recent Places
Desktop
Administrator
Computer
Network
Name Date modified Type Size
bin 15.02.2009 20:13 File Folder
Examples 15.02.2009 20:13 File Folder
Help 15.02.2009 20:13 File Folder
Icons 15.02.2009 20:13 File Folder
include 15.02.2009 20:46 File Folder
Lang 15.02.2009 20:13 File Folder
lib 15.02.2009 20:46 File Folder
libexec 15.02.2009 20:13 File Folder
mingw32 15.02.2009 20:13 File Folder
Packages 15.02.2009 20:15 File Folder
temp 16.02.2009 22:00 File Folder
Templates 15.02.2009 20:13 File Folder
File name: Project1.dev
Save as type: Dev-C++ project (*.dev)
Save
Cancel
Compiler Resources Compile Log Debug Find Results
Ready.


Dev-C++ 4.9.9.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Classes Debug
Create new project
Save in: Dev-Cpp
Recent Places
Desktop
Administrator
Computer
Network
Name Date modified Type Size
bin 15.02.2009 20:13 File Folder
Examples 15.02.2009 20:13 File Folder
Help 15.02.2009 20:13 File Folder
Icons 15.02.2009 20:13 File Folder
include 15.02.2009 20:46 File Folder
Lang 15.02.2009 20:13 File Folder
lib 15.02.2009 20:46 File Folder
libexec 15.02.2009 20:13 File Folder
mingw32 15.02.2009 20:13 File Folder
Packages 15.02.2009 20:15 File Folder
temp 16.02.2009 22:00 File Folder
Templates 15.02.2009 20:13 File Folder
File name: Project1.dev
Save as type: Dev-C++ project (*.dev)
Save
Cancel
Compiler Resources Compile Log Debug Find Results
Ready.

C/C++ D ümü



Adım 2: Kaynak dosyasının oluşturulması/eklenmesi (Create/add source file(s)).

Şimdi içeriği boş ve isimsiz kaynak dosyasına sahip projesine sahipsiniz. Basit programlar yazıyorsanız standart kütüphaneler yeterli olacaktır. Ek kaynak kodu dosyaları projenize şu şekilde eklenebilir:

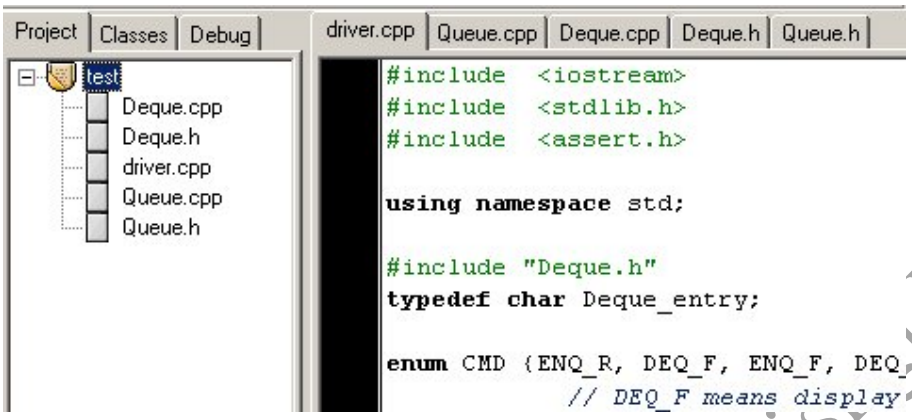
- "`File`" menüsünden "`New Source File`" (veya CTRL+U ya basınız) veya
- "`Project`" menüsünden "`New Unit in Project`" (veya CTRL+F1 e basınız). Dev-C++ size aşağıdaki işlemlere başlayana kadar birşey sormayacaktır:
    1. Derleme-Compile
    2. Projenin kayıt edilmesi-Save the project
    3. Kaynak kodun saklanması-Save the source file
    4. Dev-C++ dan çıkarken-Exit Dev-C++

    Var olan kaynak dosyasını iki şekilde projenize eklenebilir:

- "`Project`" menüsünden "`Add to project...`" (veya CTRL+F2 ye basınız) veya
- Sol taraftaki panel içindeki proje adının üzerine fare işaretçisini getiriniz ve farenin sağ tuşuna tıklayınız "`Add to project...`" (veya CTRL+F2 ye basınız).

Örnek olarak aşağıdaki şekilde çoklu kaynak dosyası projeye eklenmektedir:

C/C++ Ders Notları A.Ü. Fizik Mühendisliği Bölümü



Bu örnekte 3 dosya “driver.cpp”, "Queue.cpp" nin kullanacağı içerik dosyası "Queue.h", "Stack.cpp" dosyasının kullanacağı "Stack.h" içerik dosyaları projeye eklenmektedir.

Dikkat ederseniz Dev-C++ "`.cc`" dosya uzantısı yerine "`.cpp`" dosya uzantısını kullanmaktadır.

Adım 3: Derlemek (Compile).

Kaynak kodlarınızı ve diğer dosyaları projenize ekledikten sonra projenizi derleyebilirsiniz.

- "`Execute`" menüsünden "`Compile`" (veya CTRL+F9 a basınız). Bu aşamada derleyici size bazı uyarılar ve hataları pencerenin alt kısmında görülebilir.

  It is likely that you will get some kind of compiler or linker error the first time you attempt to compile a project. Syntax errors will be displayed in the "`Compiler`" tab at the bottom of the screen. You can double-click on any error to take you to the place in the source code where it occurred. The "`Linker`" tab will flash if there are any linker errors. Linker errors are generally the result of syntax errors not allowing one of the files to compile.

C/C++ Ders Notları A.Ü. Fizik Mühendisliği Bölümü