BURSA TEKNİK ÜNİVERSİTESİ

MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ ALGORİTMALAR VE PROGRAMLAMA DERSİ DÖNEM PROJESİ RAPORU

PROJE ADI: UZAY SİMÜLASYONU VE FİZİK DENEYLERİ

HAZIRLAYAN: Sümeyye Şimşek – 24360859058

Şube-2

OCAK 2026 BURSA

1.Giriş

1.1 Projenin Amaç ve Kapsamı

Bu proje, Bursa Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü, Algoritmalar ve Programlama dersi 2025-2026 Güz Dönemi kapsamında, 1. sınıf öğrencisi Sümeyye Şimşek tarafından dönem projesi olarak oluşturulmuştur.

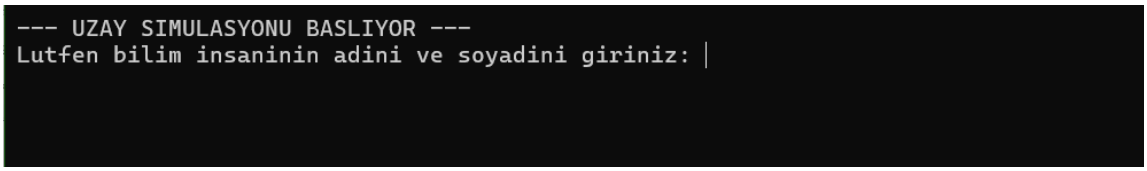
Bu proje tamamen bireysel olarak geliştirilmiştir.

Projenin amacı, C programlama dili kullanılarak konsol tabanlı bir uzay simülasyonu oluşturmaktır. Oluşturulan simülasyon, Güneş Sistemindeki farklı gezegenlerde temel fizik deneylerini kolay bir şekilde simüle edilmesini sağlar. Başlangıç adımı olarak bir bilim insanının (kullanıcının) adını girmesini ister **(Bkz. Şekil 1-3).** Ardından kullanıcıya, içerisinde Serbest Düşme, Sarkaç, Asansör Deneyi gibi 9 farklı fiziksel deneyin bulunduğu bir menü listelenir ve bir seçim yapması istenir. Kullanıcının yaptığı seçim sonrasında, program ilgili deney için gerekli metrik verileri (kütle, hız, süre vb.) ister. Kullanıcı verileri girdikten sonra uygulama arka planda her gezegenin kendine özgü yerçekimi ivmesi ile hesaplamaları yapar ve sonuçları tüm gezegenler için sırasıyla ekrana yazdırır.

```c
printf("--- UZAY SIMULASYONU BASLIYOR ---\n");
printf("Lutfen bilim insaninin adini ve soyadini giriniz: ");

scanf(" %[^\n]", bilimInsani);
```

Şekil 1: Program Açılışı ve Kullanıcı Adı Giriş Ekranı (Kod Kesiti)



Şekil 2: Program Açılışı

Şekil 3: Kullanıcı Adı Giriş Ekranı

2. Teknik Detaylar

2.1.Program Akışı

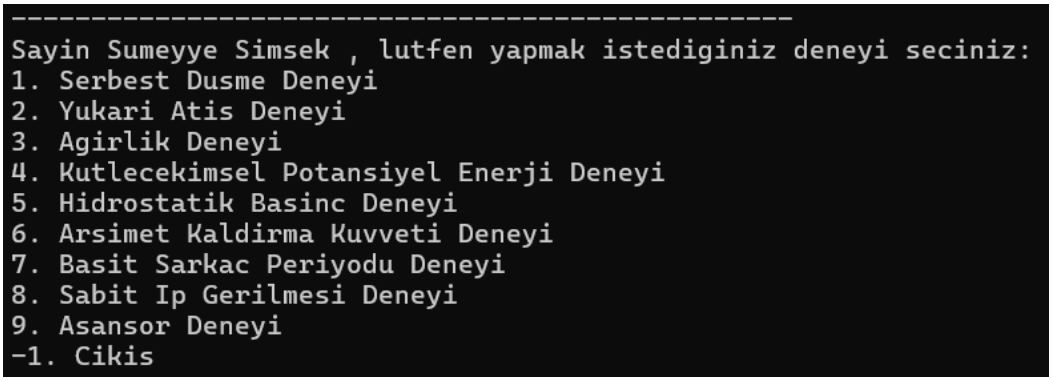
Bu proje, kod karmaşasını önlemek için Yapısal Programlama kurallarına göre modüler bir yapıda hazırlanmıştır. Her bir fizik deneyini (Serbest Düşme, Yukarı Atış, Sarkaç vb.) ana fonksiyona yığmak yerine, kodlar parçalara bölünmüştür, kendi özel (void) fonksiyonlara ayrılmış bir düzende yazılmıştır. Bu sayede kod daha okunabilir hale gelmiştir ve hataları tespit etmemizi (debug) kolaylaştırmıştır. Ayrıca bu yapı bu projeyi ilerde büyütmek istersek yeni özellikler eklememize de kolaylaştırmaktadır.

Programın ana giriş noktası olan main fonksiyonu, projenin yönetim merkezi olarak çalışmaktadır. Bu fonksiyon, herhangi bir hesaplama işlemi yapmaz; görevi sadece temel değişkenleri tanımlamak, kullanıcıdan isim bilgisini almak ve menü döngüsünü kontrol etmektir. Bu sayede, programın akış kontrolü ile hesaplama işlemleri birbirlerinden net bir şekilde ayrılmıştır. Tüm deney fonksiyonları, yer çekimi dizisini ve gezegen isimlerini parametre olarak alacak şekilde standart bir yapıda tasarlanmıştır.

Programın kullanıcı arayüzü, Menü Yönlendirmeli (Menu-Driven) bir mantık esas alınarak oluşturulmuştur. Programın döngü yapısında do-while döngüsü kullanılmıştır. While yerine do-while kullanılmasının temel sebebi, program çalışır çalışmaz kontrol yapmadan önce menüyü kullanıcıya en az bir kez menünün gösterilmesini sağlamaktır. Kullanıcı deney seçim ekranında kaldığı ve -1 girmediği sürece döngü aktif kalır ve kesintisiz devam eder. Seçim işlemleri daha düzenli olması için if-else blokları yerine switch-case yapısı kullanılmıştır. Kullanıcının yaptığı seçime göre hangi deney seçildiyse, o deneyin fonksiyonu devreye girer **(Bkz. Şekil 4 ve 5)**

```
do {
    printf("\n-----------------------------------------------------\n");
    printf("Sayin %s, lutfen yapmak istediginiz deneyi seciniz:\n", bilimInsani);
    printf("1. Serbest Dusme Deneyi\n");
    printf("2. Yukari Atis Deneyi\n");
    printf("3. Agirlik Deneyi\n");
    printf("4. Kutlecekimsel Potansiyel Enerji Deneyi\n");
    printf("5. Hidrostatik Basinc Deneyi\n");
    printf("6. Arsimet Kaldirma Kuvveti Deneyi\n");
    printf("7. Basit Sarkac Periyodu Deneyi\n");
    printf("8. Sabit Ip Gerilmesi Deneyi\n");
    printf("9. Asansor Deneyi\n");
    printf("-1. Cikis\n");
    printf("\n\n");
    printf("Seciminiz: ");
    scanf("%d", &secim);
    printf("-----------------------------------------------------\n");
```

Şekil 4: Kullanıcı Deney Seçim Menüsü ve Program Döngüsü (Kod Kesiti)



Şekil 5: Kullanıcı Deney Seçim Menüsü ve Program Döngüsü

2.2. Gezegen Verileri ve Kullanılan Sabitler

Projenin fiziksel gerçekliğe uygun olması amacıyla, Güneş Sistemi'nde yer alan 8 temel gezegenin (Merkür, Venüs, Dünya, Mars, Jüpiter, Satürn, Uranüs ve Neptün) yerçekimi ivmeleri sabit (const) değerler halinde projeye tanımlanmıştır. Fiziksel hesaplamalarda hassasiyeti artırmak ve dolayısıyla hata paylarını en aza indirmek için standart float veri tipi yerine, 64-bit hassasiyetine sahip double veri tipi kullanılmıştır. Bu veriler g_ivmeleri dizisine atanmıştır. Gezegen isimleri de char türünde iki boyutlu bir dizi içinde indeks sırasına göre saklanmaktadır; böylece hesaplama sonuçları ekrana yazdırılırken verilerin hangi gezegene ait olduğu net bir şekilde belirtilebilmektedir.

Teknik şartnamede vurgulanan 'dizilere işaretçi (pointer) ile erişim' maddesine tam uyum sağlamak amacıyla, C dilinin bellek yönetimindeki verimliliğini göstermek için, dizilerle ilgili tüm işlemlerde klasik yöntemler yerine işaretçi (pointer) yapısı tercih edilmiştir. Deney fonksiyonlarına dizilerin kendisi veya kopyası değil, bellekteki başlangıç adreslerini (base address) temsil eden pointerlar parametre olarak gönderilmiştir. Fonksiyon çağrılarında dizilerin bellek adresleri parametre olarak kullanılarak, dizinin tamamının kopyalanması engellenmiştir. Bu yöntem, bellek yönetimi açısından en verimli yol olan 'Pass by Reference' tekniğinin projeye etkin bir şekilde uygulanmasını sağlamıştır.

Fonksiyon içindeki hesaplamalarda, yaygın dizi[i] (indeksleme) yöntemi yerine, bellek adresleri üzerinde aritmetik işlem yapma (pointer arithmetic) yöntemi kullanılmıştır. Yani *(g_ptr + i) yazıldıktan sonra bellekten taban adresten i kadar öteye gidilmiş ve 'dereferencing' (*) operatörüyle o adresteki veri çekilmiştir **(Bkz. Şekil 6).** Bu sayede, dizilerin bellekte ardışık

bloklar halinde tutulduğu mantığı uygulamaya geçirilmiş; adres aritmetiği ve bellek yönetimi konusundaki teorik bilgiler projede somutlaştırılmıştır.

```
printf("\n--- Serbest Dusme Sonuclari ---\n");
for(int i=0;i<8;i++){
    // Dizi elemanlarına pointer aritmetiği ile ulaşılmıştır *(g_ptr+i)
    double g=*(g_ptr+i);
    h=0.5*g*t*t;
    printf("%-10s gezegeninde dusulen yol (h): %.2f m\n",*(isimler+i),h);
}
```

Şekil 6: Dizi Elemanlarına Pointer Aritmetiği ile Erişim (Kod Kesiti)

2.3. Deneylerin Hesaplama Mantığı

Geliştirilen projenin asıl amacı projede yer alan simülasyonların, teorik fizik yasalarının algoritmik bir düzlemde modellenerek, bilgisayar ortamında test edilebilir ve çalıştırılabilir hale getirilmesidir. Her bir deney fonksiyonu, ilgili fiziksel olayın (hareket, enerji, basınç vb.) matematiksel modeline tam uyum sağlayacak şekilde kodlanmıştır. Bu hesaplama sürecinde oluşabilecek yuvarlama hatalarını azaltmak için ve işlem hassasiyetini en üstte tutmak için tum sayısal değişkenlerde double veri tipi tercih edilmiştir. Aşağıda, projenin içeriğini oluşturan 9 farklı deneyin hesaplama yöntemleri, veri yapıları ve ilgili formülleri aşağıdaki başlıklarda teknik detaylarıyla açıklanmıştır.

2.3.1. Serbest Düşme Deneyi Bu deneyde, hava sürtünmesi ihmal edilerek bir cismin belirli bir süre boyunca yerçekimi etkisinde kat edeceği mesafe hesaplanır.

- **Alınan Girdiler:** Düşüş süresi (t) saniye cinsinden istenir.
- **Kullanılan Formül:** $\boldsymbol{h = \frac{1}{2} g t^2}$
- **Çıktı Birimi:** Metre (m). Simülasyon, girilen süreyi formülde işleyerek her gezegen için düşülen mesafeyi ayrı ayrı hesaplar **(Bkz. Şekil 7 ve 8).**

```c
// 1. Serbest Düşme
void serbestDusme(double*g_ptr, char(*isimler)[15]){
    double t,h;
    printf("Dusme suresini (t) saniye cinsinden giriniz: ");
    scanf("%lf",&t);

    // Negatif kontrolü Ternary Operator ile yapılmıştır
    t=(t<0)?-t:t;

    printf("\n--- Serbest Dusme Sonuclari ---\n");
    for(int i=0;i<8;i++){
        // Dizi elemanlarına pointer aritmetiği ile ulaşılmıştır *(g_ptr+i)
        double g=*(g_ptr+i);
        h=0.5*g*t*t;
        printf("%-10s gezegeninde dusulen yol (h): %.2f m\n",*(isimler+i),h);
    }
}
```

Şekil 7: Serbest Düşme Deneyi Fonksiyonunun Algoritma Yapısı ve Hesaplama Döngüsü (Kod Kesiti)

```
Dusme suresini (t) saniye cinsinden giriniz: 10

--- Serbest Dusme Sonuclari ---
Merkur     gezegeninde dusulen yol (h): 185.00 m
Venus      gezegeninde dusulen yol (h): 443.50 m
Dunya      gezegeninde dusulen yol (h): 490.00 m
Mars       gezegeninde dusulen yol (h): 185.50 m
Jupiter    gezegeninde dusulen yol (h): 1239.50 m
Saturn     gezegeninde dusulen yol (h): 522.00 m
Uranus     gezegeninde dusulen yol (h): 434.50 m
Neptun     gezegeninde dusulen yol (h): 557.50 m
```

Şekil 8: Serbest Düşme Deneyi Sonuçlarının Gezegenlere Göre Listelenmesi

2.3.2. Yukarı Atış Deneyi Cismin dikey olarak yukarı fırlatıldığı ve maksimum yüksekliğe ne kadar çıkabileceğinin hesaplandığı deneydir.

- **Alınan Girdiler:** Fırlatma hızı (v_0) m/s cinsinden istenir.
- **Kullanılan Formül:** $\boldsymbol{hmax = \frac{v0^2}{2g}}$
- **Çıktı Birimi:** Metre (m) **(Bkz. Şekil 9 ve 10).**

```c
// 2. Yukarı Atış
void yukariAtis(double *g_ptr, char (*isimler)[15]){
    double v0, h_max;
    printf("Firlatma hizini (v0) m/s cinsinden giriniz: ");
    scanf("%lf",&v0);

    // Hızın karesi alındığı için sonuç değişmese de, ternary ile ek kontrol sağlanmıştır
     v0=(v0<0)?-v0:v0;

    printf("\n--- Yukari Atis Sonuclari ---\n");
    for(int i=0;i<8;i++){
        double g =*(g_ptr+i);
        h_max =(v0*v0)/(2*g);
        printf("%-10s gezegeninde max yukseklik (h_max): %.2f m\n", *(isimler + i),h_max);
    }
}
```

Şekil 9: Yukarı Atış Deneyi Fonksiyonunun Algoritma ve Kod Yapısı

```
Firlatma hizini (v0) m/s cinsinden giriniz: 10

--- Yukari Atis Sonuclari ---
Merkur     gezegeninde max yukseklik (h_max): 13.51 m
Venus      gezegeninde max yukseklik (h_max): 5.64 m
Dunya      gezegeninde max yukseklik (h_max): 5.10 m
Mars       gezegeninde max yukseklik (h_max): 13.48 m
Jupiter    gezegeninde max yukseklik (h_max): 2.02 m
Saturn     gezegeninde max yukseklik (h_max): 4.79 m
Uranus     gezegeninde max yukseklik (h_max): 5.75 m
Neptun     gezegeninde max yukseklik (h_max): 4.48 m
```

Şekil 10: Yukarı Atış Deneyi Girdi ve Sonuç Çıktıları

2.3.3. Ağırlık Deneyi Bir cismin kütlesine etki eden yerçekimi kuvvetinin (ağırlık) hesaplanmasını içerir.

- **Alınan Girdiler:** Cismin kütlesi (*m*) kilogram cinsinden istenir.
- **Kullanılan Formül:** $\boldsymbol{G = mg}$
- **Çıktı Birimi:** Newton (*N*) **(Bkz. Şekil 11 ve 12).**

```
// 3. Ağırlık Deneyi
void agirlikHesapla(double *g_ptr, char (*isimler)[15]) {
    double m, G;
    printf("Cismin kutlesini (m) kg cinsinden giriniz: ");
    scanf("%lf", &m);

    //Negatif kütle kontrolü ternary ifadesi ile sağlanmıştır
    m=(m<0)?-m:m;

    printf("\n--- Agirlik Sonuclari ---\n");
    for(int i=0;i<8;i++){
        double g=*(g_ptr+i);
        G=m*g;
        printf("%-10s gezegeninde agirlik (G): %.2f Newton\n", *(isimler +i),G);
    }
}
```

Şekil 11: Ağırlık Deneyi Simülasyon (Kod Kesiti)

```
Cismin kutlesini (m) kg cinsinden giriniz: 10

--- Agirlik Sonuclari ---
Merkur     gezegeninde agirlik (G): 37.00 Newton
Venus      gezegeninde agirlik (G): 88.70 Newton
Dunya      gezegeninde agirlik (G): 98.00 Newton
Mars       gezegeninde agirlik (G): 37.10 Newton
Jupiter    gezegeninde agirlik (G): 247.90 Newton
Saturn     gezegeninde agirlik (G): 104.40 Newton
Uranus     gezegeninde agirlik (G): 86.90 Newton
Neptun     gezegeninde agirlik (G): 111.50 Newton
```

Şekil 12: Ağırlık Deneyi Simülasyon Sonuçları

2.3.4. Kütleçekimsel Potansiyel Enerji Deneyi Cismin bulunduğu yükseklik ve kütlesine bağlı olarak sahip olduğu enerjiyi simüle eder.

- **Alınan Girdiler:** Kütle (*m*) kg cinsinden ve Yükseklik (*h*) metre cinsinden istenir.
- **Kullanılan Formül:** $Ep = \boldsymbol{mgh}$
- **Çıktı Birimi:** Joule (*J*) **(Bkz. Şekil 13 ve 14).**

```c
// 4. Kütleçekimsel Potansiyel Enerji
void potansiyelEnerji(double*g_ptr, char(*isimler)[15]){
    double m,h,Ep;
    printf("Cismin kutlesini (m) kg cinsinden giriniz: ");
    scanf("%lf",&m);
    printf("Yuksekligi (h) metre cinsinden giriniz: ");
    scanf("%lf",&h);

    m=(m<0)?-m:m;
    h=(h<0)?-h:h;

    printf("\n--- Potansiyel Enerji Sonuclari ---\n");
    for(int i=0;i<8;i++) {
        double g=*(g_ptr+i);
        Ep=m*g*h;
        printf("%-10s gezegeninde enerji (Ep): %.2f Joule\n", *(isimler + i),Ep);
    }
}
```

Şekil 13: Potansiyel Enerji Hesaplama Fonksiyonu (Kod Kesiti)

```
Cismin kutlesini (m) kg cinsinden giriniz: 10
Yuksekligi (h) metre cinsinden giriniz: 20

--- Potansiyel Enerji Sonuclari ---
Merkur     gezegeninde enerji (Ep): 740.00 Joule
Venus      gezegeninde enerji (Ep): 1774.00 Joule
Dunya      gezegeninde enerji (Ep): 1960.00 Joule
Mars       gezegeninde enerji (Ep): 742.00 Joule
Jupiter    gezegeninde enerji (Ep): 4958.00 Joule
Saturn     gezegeninde enerji (Ep): 2088.00 Joule
Uranus     gezegeninde enerji (Ep): 1738.00 Joule
Neptun     gezegeninde enerji (Ep): 2230.00 Joule
```

Şekil 14: Potansiyel Enerji Deneyi Girdi ve Sonuç Ekranı

2.3.5. Hidrostatik Basınç Deneyi Sıvıların belirli bir derinlikte yüzeye uyguladığı basıncı hesaplar.

- **Alınan Girdiler:** Sıvı yoğunluğu (ρ) kg/m³ ve Derinlik (h) metre cinsinden istenir.
- **Kullanılan Formül:** $\boldsymbol{P = \rho g h}$
- **Çıktı Birimi:** Pascal (Pa) **(Bkz. Şekil 15 ve 16).**

```c
// 5. Hidrostatik Basınç
void hidrostatikBasinc(double *g_ptr, char (*isimler)[15]) {
    double ro,h,P;
    printf("Sivinin yogunlugunu (rho) kg/m^3 cinsinden giriniz: ");
    scanf("%lf",&ro);
    printf("Derinligi (h) metre cinsinden giriniz: ");
    scanf("%lf",&h);

    ro=(ro<0)?-ro:ro;
    h=(h<0)?-h:h;

    printf("\n--- Hidrostatik Basinc Sonuclari ---\n");
    for(int i=0;i<8;i++) {
        double g=*(g_ptr+i);
        P=ro*g*h;
        printf("%-10s gezegeninde basinc (P): %.2f Pascal\n", *(isimler+i),P);
    }
}
```

Şekil 15: Hidrostatik Basınç Hesaplama Algoritması (Kod Kesiti)

```
Sivinin yogunlugunu (rho) kg/m^3 cinsinden giriniz: 10
Derinligi (h) metre cinsinden giriniz: 20

--- Hidrostatik Basinc Sonuclari ---
Merkur     gezegeninde basinc (P): 740.00 Pascal
Venus      gezegeninde basinc (P): 1774.00 Pascal
Dunya      gezegeninde basinc (P): 1960.00 Pascal
Mars       gezegeninde basinc (P): 742.00 Pascal
Jupiter    gezegeninde basinc (P): 4958.00 Pascal
Saturn     gezegeninde basinc (P): 2088.00 Pascal
Uranus     gezegeninde basinc (P): 1738.00 Pascal
Neptun     gezegeninde basinc (P): 2230.00 Pascal
```

Şekil 16: Farklı Gezegenler İçin Hidrostatik Basınç Simülasyonu Çıktıları

2.3.6. Arşimet Kaldırma Kuvveti Deneyi Sıvı içerisindeki cisme etki eden kaldırma kuvvetini analiz eder.

- **Alınan Girdiler:** Sıvı yoğunluğu (ρ) ve Batan Hacim (V) istenir.
- **Kullanılan Formül:** $\boldsymbol{Fk = \rho gV}$
- **Çıktı Birimi:** Newton (N) **(Bkz. Şekil 17 ve 18).**

```c
// 6. Arşimet Kaldırma Kuvveti
void arsimetKaldirma(double *g_ptr, char (*isimler)[15]){
    double ro,V,Fk;
    printf("Sivinin yogunlugunu (rho) kg/m^3 cinsinden giriniz: ");
    scanf("%lf",&ro);
    printf("Batan hacmi (V) m^3 cinsinden giriniz: ");
    scanf("%lf",&V);

    ro=(ro<0)?-ro:ro;
    V=(V<0)?-V:V;

    printf("\n--- Kaldirma Kuvveti Sonuclari ---\n");
    for(int i=0;i<8;i++){
        double g = *(g_ptr + i);
        Fk=ro*g*V;
        printf("%-10s gezegeninde kaldirma kuvveti (Fk): %.2f Newton\n", *(isimler+i),Fk);
    }
}
```

Şekil 17: Arşimet Kaldırma Kuvveti Fonksiyonunun Kod Yapısı

```
Sivinin yogunlugunu (rho) kg/m^3 cinsinden giriniz: 10
Batan hacmi (V) m^3 cinsinden giriniz: 10

--- Kaldirma Kuvveti Sonuclari ---
Merkur     gezegeninde kaldirma kuvveti (Fk): 370.00 Newton
Venus      gezegeninde kaldirma kuvveti (Fk): 887.00 Newton
Dunya      gezegeninde kaldirma kuvveti (Fk): 980.00 Newton
Mars       gezegeninde kaldirma kuvveti (Fk): 371.00 Newton
Jupiter    gezegeninde kaldirma kuvveti (Fk): 2479.00 Newton
Saturn     gezegeninde kaldirma kuvveti (Fk): 1044.00 Newton
Uranus     gezegeninde kaldirma kuvveti (Fk): 869.00 Newton
Neptun     gezegeninde kaldirma kuvveti (Fk): 1115.00 Newton
```

Şekil 18: Arşimet Kaldırma Kuvveti Hesaplama Sonuçları

2.3.7. Basit Sarkaç Periyodu Deneyi İpin ucuna bağlı bir cismin salınım süresini (periyodunu) hesaplar. Karekök işlemi için <math.h> kütüphanesindeki sqrt() fonksiyonu kullanılmıştır.

- **Alınan Girdiler:** İp uzunluğu (L) metre cinsinden istenir.
- **Kullanılan Formül:** $\boldsymbol{T = 2\,\pi\,\sqrt{(\frac{L}{g})}}$
- **Çıktı Birimi:** Saniye (s) **(Bkz. Şekil 19 ve 20).**

```c
// 7. Basit Sarkaç
void basitSarkac(double *g_ptr, char (*isimler)[15]) {
    double L,T;
    printf("Sarkacin uzunlugunu (L) metre cinsinden giriniz: ");
    scanf("%lf",&L);

    L=(L<0)?-L:L;

    printf("\n--- Basit Sarkac Periyodu Sonuclari ---\n");
    for(int i=0; i<8;i++){
        double g=*(g_ptr+i);
        // T=2*pi*sqrt(L / g)
        T=2*PI*sqrt(L/g);
        printf("%-10s gezegeninde periyot (T): %.2f saniye\n",*(isimler+i),T);
    }
}
```

Şekil 19: Sarkaç Periyodu Hesaplama Fonksiyonu ve Math Kütüphanesi Kullanımı (Kod Kesiti)

```
Sarkacin uzunlugunu (L) metre cinsinden giriniz: 10

--- Basit Sarkac Periyodu Sonuclari ---
Merkur     gezegeninde periyot (T): 10.33 saniye
Venus      gezegeninde periyot (T): 6.67 saniye
Dunya      gezegeninde periyot (T): 6.35 saniye
Mars       gezegeninde periyot (T): 10.32 saniye
Jupiter    gezegeninde periyot (T): 3.99 saniye
Saturn     gezegeninde periyot (T): 6.15 saniye
Uranus     gezegeninde periyot (T): 6.74 saniye
Neptun     gezegeninde periyot (T): 5.95 saniye
```

Şekil 20: Basit Sarkaç Deneyi Karşılaştırmalı Sonuç Listesi

2.3.8. Sabit İp Gerilmesi Deneyi Sabit duran ve bir ipe asılı olan cisme etki eden gerilme kuvvetini simüle eder.

- **Alınan Girdiler:** Asılı cismin kütlesi (*m*) istenir.
- **Kullanılan Formül:** $\boldsymbol{T = mg}$

- **Çıktı Birimi:** Newton (*N*) **(Bkz. Şekil 21 ve 22).**

```c
// 8. Sabit İp Gerilmesi
void ipGerilmesi(double *g_ptr, char (*isimler)[15]) {
    double m, T;
    printf("Asili cismin kutlesini (m) kg cinsinden giriniz: ");
    scanf("%lf", &m);

    m=(m<0)?-m:m;

    printf("\n--- Ip Gerilmesi Sonuclari ---\n");
    for(int i=0;i<8;i++) {
        double g=*(g_ptr + i);
        T=m*g; // F = m*g
        printf("%-10s gezegeninde ip gerilmesi (T): %.2f Newton\n", *(isimler+i),T);
    }
}
```

Şekil 21: Sabit İp Gerilmesi Fonksiyonu (Kod Kesiti)

```
Asili cismin kutlesini (m) kg cinsinden giriniz: 10

--- Ip Gerilmesi Sonuclari ---
Merkur     gezegeninde ip gerilmesi (T): 37.00 Newton
Venus      gezegeninde ip gerilmesi (T): 88.70 Newton
Dunya      gezegeninde ip gerilmesi (T): 98.00 Newton
Mars       gezegeninde ip gerilmesi (T): 37.10 Newton
Jupiter    gezegeninde ip gerilmesi (T): 247.90 Newton
Saturn     gezegeninde ip gerilmesi (T): 104.40 Newton
Uranus     gezegeninde ip gerilmesi (T): 86.90 Newton
Neptun     gezegeninde ip gerilmesi (T): 111.50 Newton
```

Şekil 22: Sabit İp Gerilmesi Deneyi Simülasyon Çıktıları

2.3.9. Asansör Deneyi İvmelenen bir asansör içindeki cismin "hissedilen ağırlığını" (Görünür Ağırlık) hesaplar.

- **Alınan Girdiler:** Kütle (m), Asansör İvmesi (a) ve Hareket Yönü istenir.
- **Kullanılan Formül:**
  - Yukarı Hızlanan / Aşağı Yavaşlayan: $\boldsymbol{N = m(g + a)}$
  - Aşağı Hızlanan / Yukarı Yavaşlayan: $\boldsymbol{N = m(g - a)}$
- **Çıktı Birimi:** Newton (N) **(Bkz. Şekil 23-25).**

```c
// 9. Asansör Deneyi
void asansorDeneyi(double *g_ptr, char (*isimler)[15]) {
    double m,a,N;
    int durum;

    printf("Cismin kutlesini (m) kg cinsinden giriniz: ");
    scanf("%lf",&m);
    printf("Asansorun ivmesini (a) m/s^2 cinsinden giriniz: ");
    scanf("%lf",&a);

    m=(m<0)?-m:m;
    a=(a<0)?-a:a;

    printf("Asansorun hareket durumunu seciniz:\n");
    printf("1. Yukari Hizlanan VEYA Asagi Yavaslayan (mg + ma)\n");
    printf("2. Asagi Hizlanan VEYA Yukari Yavaslayan (mg - ma)\n");
    printf("Secim: ");
    scanf("%d", &durum);

    printf("\n--- Asansor Deneyi Sonuclari ---\n");
    for(int i=0;i<8;i++){
        double g=*(g_ptr+i);
```

Şekil 23: Asansör Deneyi İvme ve Yön Seçim Fonksiyonunun Mantıksal Yapısı (Kod Kesiti 1)

```c
    if (durum==1) {
        // N=m(g+a)
        N=m*(g+a);
    } else {
        // N=m(g-a)
        N=m*(g-a);
        // Eğer ivme yerçekiminden büyükse ve aşağı hızlanıyorsa ip gevşeyebilir ama
        // negatif sonuç çıkarsa mutlak değer alabiliriz veya olduğu gibi bırakırız.
        // Fiziksel olarak yüzey tepkisi negatif olamaz (havada kalır),
        // ama formül gereği direkt yazdırıyorum.
    }

    printf("%-10s gezegeninde hissedilen agirlik (N): %.2f Newton\n",*(isimler + i), N);
}
```

Şekil 24: Asansör Deneyi İvme ve Yön Seçim Fonksiyonunun Mantıksal Yapısı (Kod Kesiti 2)

```
Cismin kutlesini (m) kg cinsinden giriniz: 10
Asansorun ivmesini (a) m/s^2 cinsinden giriniz: 20
Asansorun hareket durumunu seciniz:
1. Yukari Hizlanan VEYA Asagi Yavaslayan (mg + ma)
2. Asagi Hizlanan VEYA Yukari Yavaslayan (mg - ma)
Secim: 2

--- Asansor Deneyi Sonuclari ---
Merkur     gezegeninde hissedilen agirlik (N): -163.00 Newton
Venus      gezegeninde hissedilen agirlik (N): -111.30 Newton
Dunya      gezegeninde hissedilen agirlik (N): -102.00 Newton
Mars       gezegeninde hissedilen agirlik (N): -162.90 Newton
Jupiter    gezegeninde hissedilen agirlik (N): 47.90 Newton
Saturn     gezegeninde hissedilen agirlik (N): -95.60 Newton
Uranus     gezegeninde hissedilen agirlik (N): -113.10 Newton
Neptun     gezegeninde hissedilen agirlik (N): -88.50 Newton
```

Şekil 25: Asansör Deneyi İçin Hissedilen Ağırlık Sonuçları

2.4. Girdi Doğrulama ve Hata Yönetimi

Programın kararlı çalışması için iki temel kontrol mekanizması geliştirilmiştir:

2.4.1 Negatif Değer Kontrolü

Fiziksel simülasyonların tutarlılığı ve gerçekçiliği açısından, kullanıcıdan alınan girdilerin (kütle, zaman, uzunluk vb.) negatif olmaması gerekmektedir. Örnek olarak negatif bir kütle veya zaman değeri fiziksel gerçeklikle örtüşmez. Sisteme yanlışlıkla girilebilecek negatif değerlere karşı önlem alınmıştır. Programın aniden kapanması (crash) önlemek ve şartnamede belirtilen "if-else yapısı yerine **Ternary Operator** kullanılmalıdır" kuralı uygulanarak etkin bir hata yönetimi uygulanmıştır. Kullanıcıdan veri alındığı anda aşağıda gösterilen satır devreye girer **(Bkz. Şekil 26).** Ve program, kullanıcının girdiği negatif değeri otomatik olarak düzeltir ve sonucu doğru (pozitif) hesaplar.

```
printf("Cismin kutlesini (m) kg cinsinden giriniz: ");
scanf("%lf",&m);
printf("Yuksekligi (h) metre cinsinden giriniz: ");
scanf("%lf",&h);

m=(m<0)?-m:m;
h=(h<0)?-h:h;
```

Şekil 26: Negatif Sayı Kontrolü (Kod Kesiti)

2.4.2 Hatalı Karakter Girişi ve Buffer

Programın çalışması için verilerin istenilen tipe uygun olması ve yanlış veri tipi girişinden kaynaklanan sonsuz döngü riskini ortadan kaldırmak amacıyla, giriş tamponu **(buffer)** kontrol mekanizması eklenmiştir. Scanf fonksiyonunun dönüş değeri denetlenerek harf veya sembol girilmesi durumunda bellekte kalan hatalı veriler temizlenmekte ve sistem kilitlenmeden kullanıcıdan tekrar giriş yapması sağlanmaktadır. **(Bkz. Görsel 27)**

```
// Tek bir scanf ile veri hem okuyor hem kontrol ediliyor.
// Eğer sayı girilmezse (harf yada sembol girilirse) if'in içine girip temizliyor
if (scanf("%d", &secim) != 1) {
    while(getchar() != '\n'); // Tampon belleği temizle (enter'a kadar olan harfleri yutar)
    secim = 0; // Programın kırılmaması için geçersiz bir sayı verilmiştir
}
```

Şekil 27: Hatalı Karakter Girişi Kontrolü

3. EKSİKLİKLER VE GELİŞTİRMELER

Projenin oluşturulma aşamasında, teknik şartnamede belirtilen kriterlerin eksiksiz karşılanması ve sistemin kararlığının korunmasına büyük önem verilmiştir. Projede öncelik temel algoritmaların sorunsuz çalışmasına verilmiştir ve yapısını sağlam bir zemine oturtmak amacıyla 'Artırımlı Geliştirme' yöntemi kullanılmıştır. Bu nedenle şu anki aşamasında kritik olmayan bazı özellikler sonraki sürümlere bırakılmıştır. Aşağıda, bu eksikliklerin nedenleri ve giderilmesi durumunda projeye sağlayacağı avantajlardan bahsedilmiştir.

3.1. Otomatik Raporlama ve Veri Dışa Aktarımı (Data Export)

- **Mevcut Kısıt (Current Limitation):** Mevcut simülasyon, hesaplanan verileri yalnızca anlık olarak ekrana (stdout) basmaktadır. C dilindeki standart giriş-çıkış fonksiyonları kullanıldığı için, program kapatıldığında RAM üzerindeki tüm deney verileri kalıcı olarak silinmektedir. Bu durum, bilim insanının geriye dönük analiz yapmasını engellemektedir. Ancak ilerleyen zamanlarda bu programla yapılan değerleri bir CSV dosyasına yazdırılarak kalıcı bir deney sistemi oluşturulabilir.
- **Sisteme Katkısı:** Bu geliştirme sayesinde, simülasyondan elde edilen veriler Excel veya MATLAB gibi harici mühendislik yazılımlarına aktarılabilecek, grafiksel analizler ve veri madenciliği işlemleri mümkün hale gelecektir. Proje, sadece hesaplayan değil, aynı zamanda "veri üreten" bir sisteme dönüşecektir.

3.2. Grafiksel Kullanıcı Arayüzü (GUI) Entegrasyonu

- **Mevcut Durum:** Projenin mevcut etkileşimi, siyah konsol ekranı (Command Prompt) üzerinden metin tabanlı olarak yürütülmektedir. Kullanıcı deneyimi, klavye girdileri ve metin çıktılarıyla sınırlıdır. Ama ilerleyen sürümlerde kullanıcı ara yüzü tasarlanırken butonlar, grafikler vb. şeylerin kullanılması hedeflenmektedir.
- **Kazanç:** Kullanıcı Deneyimi standartları modern seviyeye çekilmiş olur. Özellikle fiziksel simülasyonların görselleştirilmesi, projenin eğitim materyali olarak kullanılabilmesinin önünü açar.

3.3. Süreçte Karşılaşılan Teknik Zorluklar ve Çözümleri

Projenin oluşturulmasında en çok çaba harcanan kısım, Pointer yapısının dizilere uygulanması olmuştur. Özellikle g_ivmeleri dizisine erişirken klasik indeks ([]) yöntemi yerine adres operatörlerini (*(ptr + i)) kullanmak, başlangıçta bellek erişim hatalarına (Segmentation Fault) yol açmıştır. Bu sorunu aşmak için C dili dokümantasyonları incelenerek ve çeşitli akademik kaynaklardan faydalanılarak ve bellek adreslerinin (hexadecimal) çalışma mantığı üzerine literatür taraması yapılarak ve pointer aritmetiğinin, dizinin başlangıç adresinden veri tipi boyutu (double için 8 byte) kadar öteye gitmek anlamına geldiği kavranmıştır. Ayrıca negatif değer girişlerinde programın akışını bozmadan düzeltme yapan "Ternary Operator'ü" mantığını ve yanlış karakter girilmesi sonucu buffer temizliğinin kurgulamak, algoritmik düşünme becerisini geliştiren önemli bir süreç olmuştur.

,

4. SONUÇ

Bursa Teknik Üniversitesi "Algoritmalar ve Programlama" dersi kapsamında geliştirdiğim bu proje, derste öğrendiğimiz teorik bilgileri gerçek bir uygulamaya dönüştürmem açısından çok faydalı oldu. Amacım sadece çalışan bir kod yazmak değil, aynı zamanda okunaklı ve hatasız çalışan bir yapı kurmaktı.

Proje sürecinde en çok C Programlama dili ve özellikle pointer (işaretçi) konusunu pekiştirdim. Başlangıçta dizilere pointer aritmetiği ile (*(ptr + i)) erişmek zorlasa da bu projeyle belleğin nasıl çalıştığını ve verilerin hafızada nasıl tutulduğunu çok daha iyi anladım. Standart yöntemler yerine pointer kullanmanın mantığını kavramak benim için önemli bir tecrübe oldu.

Ayrıca fizik kurallarını kodlayarak algoritmik düşünme becerimi geliştirdim. Kullanıcının yanlışlıkla negatif değer girmesi gibi durumlarda programın çökmemesi için eklediğim kontroller ("Ternary Operator") ya da yanlış karakter girilince yapılan kontroller, bana sağlam kod yazmanın önemini gösterdi.

Sonuç olarak; hem istenen teknik şartları (pointer, modüler yapı, menü sistemi) yerine getirdim hem de bu proje sayesinde ilerideki mühendislik derslerim için sağlam bir temel atmış oldum.

5. KAYNAKÇA

Bursa Teknik Üniversitesi. (2025). *Algoritmalar ve Programlama Dersi Notları ve Sunumları*. Bilgisayar Mühendisliği Bölümü.

cppreference.com. (t.y.). *C numerics library (math.h)*. Erişim Tarihi: 5 Ocak 2026, Erişim Adresi: https://en.cppreference.com/w/c/numeric/math

Deitel, P. J., & Deitel, H. M. (2015). *C How to Program (8. Basım)*. Pearson Education.

Serway, R. A., & Jewett, J. W. (2018). *Fen ve Mühendislik İçin Fizik (Physics for Scientists and Engineers)*. Cengage Learning.

Şimşek, S. (2026). *Uzay Simülasyonu Projesi Kaynak Kodları*. GitHub. Erişim Adresi: https://github.com/sumeyyesiimsek/24360859058-SumeyyeSimsek-AvpProje