

BİLGİSAYAR MÜHENDİSLİĞİ ALGORİTMALAR VE PROGRAMLAMA PROJE ÖDEVİ

İsim: Muhammed Emin

Soyisim: Pehlivan

Öğrenci_no: 25360859021

Fakülte/Bölüm: Mühendislik ve Doğa Bilimleri Fakültesi / Bilgisayar Mühendisliği

Şube: 2. grup



Bu proje bireysel olarak geliştirilmiştir.

İçindekiler

Kapak

İçindekiler

0. Giriş
1. Teknik detaylar
1.0 Program Akışı ve Modüler Yapı).....
1.1 Gezegen Verileri ve Kullanılan Sabitler
1.2 Deneysel Hesaplama Mantığı
1.2.0 Serbest Düşme
1.2.1 Yukarı Atış
1.2.2 Ağırlık
1.2.3 Kütleçekimsel Potansiyel Enerji
1.2.4 Hidrostatik Basınç
1.2.5 Arşimet Kaldırma Kuvveti
1.2.6 Sabit İp Gerilmesi
1.2.7 Basit Sarkaç Periyodu
1.2.8 Asansör
1.3 Girdi Doğrulama ve Hata Yönetimi
2. Eksiklikler ve Geliştirmeler.....
3. Sonuç.....
4. Kaynakça.....

0. Giriş

Bu raporda bizden istenen algoritma belirli hususlar çerçevesinde anlatılacaktır. Kısaca algoritmanın çalışma şeklinden bahsetmemiz gerekirse:

Kullanıcıdan isim alınır ardından kullanıcıya terminal üzerinden 9 deney gösterilir, kullanıcının yaptığı seçime göre deney sonucunun hesaplanması için gerekli parametreler istenir, ardından her gezegen için değişen deney sonuçları kullanıcıya gösterilir.

Algoritmanın yazılması konusunda bazı şartlar da bulunur ancak bu teknik kısımlar başlığı altında incelenecaktır. Hazırsanız başlayalım.

1. Teknik kısımlar

1.0 Program Akışı ve Modüler Yapı

```
/*=====NAME_INPUT=====*/
while (1) {
    printf("isminizi girin (maksimum 49 karakter): ");
    if (!fgets(isim, sizeof(isim), stdin)) continue;

    if (! strchr(isim, '\n')) {
        printf("Karakter siniri asildi!\n");
        while (getchar() != '\n');
        continue;
    }

    isim[strcspn(isim, "\n")] = '\0';
    if (isim[0] == '\0') continue;
    break;
}
```

Şekil 1

İlk olarak bir döngü içinde kullanıcıdan şekil 1' de olduğu gibi isim alınır, eğer ki isim uygun şekilde girilmişse döngüden çıkarılır ve ikinci kısma geçilir, eğer ki isim istenildiği gibi girilmemişse hata verir ve kullanıcı uygun bir isim girene dek döngü devam eder.

```

printf("\nHosgeldin %s!\n",isim);

/*==SHOW_EXPERIMENT==*/
deney_secim();

/*==MAIN_LOOP==*/
while(1){
    double result = 0.0;
    if (!guvenli_int_al("\nBir deney seciniz: ", &choice)){
        printf("Gecersiz secim!\n");
        continue;
    }
    switch (choice) {
        case -1:printf("cikis yapiliyor...");break;
        case 1: Serbest_Dusme(&result); break;
        case 2: Yukari_Aitis(&result); break;
        case 3: Agirlik(&result); break;
        case 4: Kutlecekimsel_Potansiyel_Enerji(&result); break;
        case 5: Hidrostatik_Basinc(&result); break;
        case 6: Arsimet_Kaldirma_Kuvveti(&result); break;
        case 7: Sabit_ip_Gerilmesi(&result); break;
        //ozel durumlular
        case 8:Basit_Sarkac_Periyodu(gezegen_isim,gezegen_ivme);break;
        case 9: Asansor(gezegen_isim,gezegen_ivme);break;
        default: printf("Geçerli araligin disinda bir deger girdiniz!\n");continue;
    }
    if(choice==-1) break;
    if(choice<8) Gezegen_islem(choice,gezegen_isim,8,gezegen_ivme,result,deneys_birim);
}

```

Şekil 2

Döngüden çıktıktan sonra kullanıcı se-lamlanır ve kullanıcıya deney_secim(şekil 3) fonksiyonu ile deneyler terminalde gösterilir. Ardından ana döngüye(şekil 2) girilir. Burada kullanıcıya güvenlik fonksiyonu ve switch case yapısı ile deney seçmesi sağlanır. Seçimi

```

void deney_secim(void) {
    printf("1- Serbest Dusme\n");
    printf("2- Yukari Atis\n");
    printf("3- Agirlik\n");
    printf("4- Potansiyel Enerji\n");
    printf("5- Hidrostatik Basinc\n");
    printf("6- Arsimet Kuvveti\n");
    printf("7- ip Gerilmesi\n");
    printf("8- Basit Sarkac\n");
    printf("9- Asansor\n");
}

```

Şekil 3

uygun şekilde yaptığı takdirde hangi deneyi seçmişse o deneye ait fonksiyon çağrılır. Burada kısaca algoritmanın çalışmasından bahsetmekte yarar var çünkü bu algoritmada normale göre biraz dahamfarklı bir yöntem izlendi. Benim gördüğüm kadarıyla bu algoritmada her deney için ayrı bir fonksiyon oluşturulur ve sonuç her fonksiyonun kendisinde bulunan bir döngü ile sağlanır. Ancak bu algoritmada ise gidilen yöntemde ise şu yapıldı. Yapılan 9 deneyin hepsinde g ifadesi kullanılır ve 7 deneyde de g değeri çarpım durumundadır. Algoritma bu noktayı kullanır. 7 deney fonksiyonunda için öncelikle g_siz değer kullanıcının alınan veriler ile oluşturulur ve bu değer result içine pointer kullanılarak atanır. Artık elimizde g_siz bir değer var. Deney seçimi sonrasında bir if koşulu eğer ki seçilen deney o 7 deneyden biriyse çalışır ve içindeki Gezegen_islem fonksiyonu çalışır(if koşulu olmasının sebebi diğer iki fonksiyonun bağımsız çalışıyor olması bu yüzden Gezegen_islem fonksiyonuna girmemeliler), bu fonksiyonun amacı oldukça basittir, kullanıcı 7 deneyden birini seçmişse bu 7 deneyi aynı kalıba koyarak kullanıcıya sonucu yazdırır. Bu nasıl oluyor diye düşünebilirsiniz işte şimdi o kısma geçiyoruz.

Bunu anlayabilmek için Gezegen_islem fonksiyonuna yakından bakmalıyız. Öncelikle bu fonksiyonun parametrelerini inceleyelim:

```
void Gezegen_islem(int a,const char **isimler,const int gezegen_sayisi,const float *gezegen_g,
                    double result,const char **birimler){

    for(int i=0;i<gezegen_sayisi;i++) {
        double g = *(gezegen_g+i);
        if(a==2) g = 1.0/g;

        printf("[%7s] = %8.2lf %s\n",*(isimler+i),result * g,* (birimler+ a-1));
    }
}
```

şekil 4

Öncelikle şunu belirtmeliyim, algoritma yazarken isterlerden biri fonksiyona gönderilecek parametrelerden biri olan diziler pointer olarak gönderilmelidir. Konumuza bakarsak şekil 4'te bulunan Gezegen_islem fonksiyonundak ilk parametre olan a seçim değerimizi tutan choice değerini, **isimler gezegen_isim adındaki string dizisinin başlangıç adresini, gezegen_sayisi kaç gezegen olduğunu, *gezegen_g gezegen_ivme adındaki float dizisinin başlangıç adresini, result g_siz deney sonucunu ve **birimler ise deney_birim adındaki dizinin başlangıç adresini tutar.

Aslında çalışma mantığı son derece basittir. Fonksiyon void seçildi çünkü geriye değer döndürmeyecek sadece terminalden çıktı verecek. Fonksiyon içindeki for döngüsü ise bizim çıktı üretmemizi sağlayacak. iterasyon sayısı gezegen_sayisi parametresine bağlıdır, her iterasyonda aynı deneyin sırayla merkür, venüs, dünya, mars, jüpİtern, saturn, uranus ve neptün gezegenleri için sonucu yazdırılır. Çıktıda önce gezegen ismi, sonra result parametresinin iterasyona göre değişen g değeri ile çarpımı ve son olarak da hangi deney ise o deneyin sonucunda istenen birim **birimler parametresi ile yazdırılır. a asında bu fonksiyonun parametrelerinin deneye göre değişimini sağlayan etmendir. Bu sayede tek döngüde 7 farklı deney çıktısı değeri ve birimine kadar yazılmış olur. Burada gezegen ivmesini g adındaki bir variable içinde tutmamızın sebebi 2. deneyde gizlidir. 2. deneyde g değeri 1/g olarak çarpıldığı için doğrudan parametreyi yani pointerin üzerine çarpma göre tersini alıp işleme sokamayız çünkü bu sefer gezegen_ivme dizisindeki bütün değerler değişmiş olur bunu engellemek için lokal bir variable oluşturmak mantıklı bir fikirdir. Bu sayede döngüde if(a==2) olduğu taktirde g değeri çarpma göre tersi alınır ve işleme girer, ana diziye zarar gelmez, böylece sonraki deneyler seçildiğinde tekrar aynı dizi kullanılabilir. Diğer iki fonksiyon ise bu fonksiyona bağlı değildir kendi fonksiyonları içinde çıktıyı üretirler çünkü formülleri bu kalıba uymaz. Bu yüzden raporda normal deney başlangıçta bahsettiğimiz 7 deney, özel deney ise sondaki iki deney olarak geçer. Şimdi Fonksiyondaki sabitlere ve bildirimlere bakalım.

1.1 Gezegen Verileri ve Kullanılan Sabitler

```
#define PI 3.141592
/*==EXPERIMENT_CHOICE_FUNCT==*/
void deney_secim(void);

/*==NORMAL_FUNCT==*/
void Serbest_Dusme(double *result);
void Yukari_Aitis(double *result);
void Agirlik(double *result);
void Kutlecekimsel_Potansiyel_Enerji(double *result);
void Hidrostatik_Basinc(double *result);
void Arsimet_Kaldirma_Kuvveti(double *result);
void Sabit_ip_Gerilmesi(double *result);

/*==SPECIAL_FUNCT==*/
void Basit_Sarkac_Periyodu(const char** isimler,const float* gezegen_g);
void Asansor(const char** isimler,const float* gezegen_g);

/*==NORMAL_FUNCT_RESULT_FUNC==*/
void Gezegen_islem(int a,const char **isimler,const int gezegen_sayisi,const float *gezegen_g,
                   double result,const char **birimler);

/*=====MAIN_FUNCT=====*/
int main(void)
{
    /*=====IMPORTANT_ARRAYS_AND_VARIABLES=====*/
    const char* deney_birim[]={"m","m","N","j","N/m^2","m^3","N"};
    const char* gezegen_isim[]{"merkur","venus","dunya","mars","jupiter","saturn","uranus","neptun"};
    const float gezegen_ivme[8]={3.70,8.87,9.81,3.71,24.79,10.44,8.69,11.15};
    char isim[50];
    int choice;
```

Şekil 5

Şekil 5 aslında bu metnin anlatacağı noktadır. Pi sayısı bir deneyde formülde kullanıldığı için define olarak tanımlandı, kullanılan tüm fonksiyonlar void fonksiyondur çünkü herhangi bir değer döndürülmesine gerek yoktur. Normal fonksiyonlarda parametre olarak result değişkenin bellek adresi gönderilir bu sayede fonksiyon içinde değeri değiştirilmiş olur. Özel durumu olan deneylerde ise gezegenin ismi ve yerçekimi olan string ve float dizilerinin bellek adresleri gönderilir çünkü bu fonksiyonlarda doğrudan çıktı yazdırılacak. Normal fonksiyonlardan çıkan g_siz ifadenin çıktı haline getirilmesi için Gezegen_islem fonksiyonu bildirilmiştir. Main fonksiyona geldiğimizde ise 4 dizi ve 1 değişken bulunur; deney_birim string dizisi Gezegen_islem fonksiyonda seçilen deneye göre çıktıının birimini belirmek için, gezegen_isim string dizisi tüm deneyler için yapılacak çıktıda hangi iterasyondaki işlem sonucunun hangi gezegene ait olduğunu göstermek için ve gezegen_ivme ise her deneyde her iterasyonda g_siz olan deney sonucuna g eklemek için oluşturulmuştur. İsim dizisi kullanıcıdan isim almak için, choice değişkeni ise kullanıcidan alınan deney seçimini tutmak için oluşturulmuştur. Görselde olmasa da stdio.h stdlib.h string.h ve math.h kütühaneleri algoritmada kullanılmış, string.h güvenlik fonksiyonundaki çeşitli fonksiyonlar için, math.h ise karekök fonksiyonu için algoritmaya dahil edilmiştir. Şimdi ise deneylerin hesaplama mantığına geçelim.

1.2 Deneylerin Hesaplama Mantığı

Bu kısımda 7 normal, iki özel toplamda 9 deneyin fonksiyonlarını, veri girişlerini ve ekran çıktılarını şekiller üzerinden inceleyeceğiz. Buradaki tüm fonksiyonlar void tanımlıdır. Çünkü fonksiyondan herhangi bir değer döndürmesine gerek yoktur. Ayrıca her fonksiyonda (asansördeki a hariç) tüm girilecek değerler ternary operator ile pozitif yapılır.

1.2.0 Serbest Düşme

```
void Serbest_Dusme(double *ptr1){  
    /*      h = g*t*t/2      */  
  
    printf("Serbest dusme deneyi secildi\n");  
    double t;  
    while (!guvenli_double_al("Cismin havada kalma suresi(s): ", &t))  
        printf("Gecersiz sayi!\n");  
    (t<0) ? t*= -1 : (void) 0;  
  
    *ptr1 += t*t/2;  
}
```

Şekil 6

Serbest düşme fonksiyonu(Şekil 6) $h=g*t*t/2$ şeklinde formülize edilir. Normal fonksiyonlarda fonksiyona parametre result değişkeni değeri değişmesi için pointer olarak gönderilir. Burada g değeri Gezegen_islem fonksiyonunda eklenir dolayısıyla kullanıcıdan sadece cismin havada kalma süresi alınır, $t*t/2$ hesaplanır ve result içine eklenir.

```
isminizi girin (maksimum 49 karakter): ahmet  
Hosgeldin ahmet!  
  
1- Serbest Dusme  
2- Yukari Atis  
3- Agirlik  
4- Potansiyel Enerji  
5- Hidrostatik Basinc  
6- Arsimet Kuvveti  
7- ip Gerilmesi  
8- Basit Sarkac  
9- Asansor  
  
Bir deney seciniz: 1  
Serbest dusme deneyi secildi  
Cismin havada kalma suresi(s): 12  
[ merkur] = 266.40 m  
[ venus] = 638.64 m  
[ dunya] = 706.32 m  
[ mars] = 267.12 m  
[ jupiter] = 1784.88 m  
[ saturn] = 751.68 m  
[ uranus] = 625.68 m  
[ neptun] = 802.80 m
```

Çıktıya baktığımızda(Şekil 7); kullanıcı isim girer kullanıcı selamlanır, deney menüsü gösterilir, Serbest Dusme seçilir, cismin havada kalma süresi istenir ve girilen değere göre her gezegen için farklı h yüksekliği kullanıcıya gösterilir.

Şekil 7

1.2.1 Yukarı Atış

```
void Yukari_Atis(double *ptr1) {  
    /*      h = v*v/2g      */  
  
    printf("Yukari atis deneyi secildi\n");  
    double v_ilk;  
    while (!guvenli_double_al("Ilk hiz(m/s): ", &v_ilk))  
        printf("Gecersiz sayi!\n");  
    (v_ilk<0) ? v_ilk=-1 : (void)0;  
  
    *ptr1 += v_ilk*v_ilk/2;  
}
```

şekil 8

Yukarı atış fonksiyonu(şekil 8) normal bir foksiyondur, $h=v^*v/2g$ şeklinde formülize edilmiş deneyde kullanıcı ilk hız değerini girer, sonra pointer ile resulta g_siz sonuç yani $v_{ilk}^*v_{ilk}/2$ eklenir.

```
isminizi girin (maksimum 49 karakter): mehmet  
Hosgeldin mehmet!  
1- Serbest Dusme  
2- Yukari Atis  
3- Agirlilik  
4- Potansiyel Enerji  
5- Hidrostatik Basinc  
6- Arsimet Kuvveti  
7- ip Gerilmesi  
8- Basit Sarkac  
9- Asansor  
  
Bir deney seciniz: 2  
Yukari atis deneyi secildi  
Ilk hiz(m/s): 35  
[ merkur] = 165.54 m  
[ venus] = 69.05 m  
[ dunya] = 62.44 m  
[ mars] = 165.09 m  
[ jupiter] = 24.71 m  
[ saturn] = 58.67 m  
[ uranus] = 70.48 m  
[ neptun] = 54.93 m
```

Çıktiya baktığımızda(şekil 9); kullanıcı isim girer,kullanıcı selamlanır, deney menüsü gösterilir, Yukarı Atis deneyi seçilir, cismin yukarıya fırlatma anındaki ilk hızı istenir ve girilen değere göre her gezegen için farklı yükseklik değerleri kullanıcıya gösterilir.

şekil 9

1.2.2 Ağırlık

```
void Agirlik(double *ptr1) {
    /*      G = m*g      */
    printf("Agirlik deneyi secildi\n");
    double m;
    while (!guvenli_double_al("Kutle(kg): ", &m) )
        printf("Gecersiz sayi!\n");
    (m<0) ? m*=-1 : (void) 0;

    *ptr1 += m;
}
```

Şekil 10

Ağırlık fonksiyonu(Şekil 10) normal bir fonksiyondur. $G=m.g$ ile formüle edilen deneyde kullanıcından kütle değeri istenir ve doğrudan result içine pointer sayesinde eklenir.

```
isminizi girin (maksimum 49 karakter): selim
Hosgeldin selim!
1- Serbest Dusme
2- Yukari Atis
3- Agirlik
4- Potansiyel Enerji
5- Hidrostatik Basinc
6- Arsimet Kuvveti
7- ip Gerilmesi
8- Basit Sarkac
9- Asansor

Bir deney seciniz: 3
Agirlik deneyi secildi
Kutle(kg): 65
[ merkur] = 240.50 N
[ venus] = 576.55 N
[ dunya] = 637.65 N
[ mars] = 241.15 N
[jupiter] = 1611.35 N
[ saturn] = 678.60 N
[ uranus] = 564.85 N
[ neptun] = 724.75 N
```

Şekil 11

Cıktıya baktığımızda; kullanıcı isim girer kullanıcı selamlanır, deney menüsü gösterilir, Agirlik deneyi seçilir, cismin kütlesi istenir ve girilen değere göre her gezegen için cismin ağırlık değeri kullanıcıya gösterilir.

1.2.3 Kütleçekimsel Potansiyel enerji

```
void Kutlecekimsel_Potansiyel_Enerji(double *ptr1){  
    /*      E = m*g*h      */  
  
    printf("Kutlecekimsel potansiyel enerji deneyi secildi\n");  
    double m,h;  
    while (!guvenli_double_al("Kutle(kg): ", &m)) printf("Gecersiz!\n");  
    (m<0) ? m*=-1 : (void)0;  
    while (!guvenli_double_al("Yukseklik(m): ", &h)) printf("Gecersiz!\n");  
    (h<0) ? h*=-1 : (void)0;  
  
    *ptr1 += m*h;  
}
```

şekil 12

Kütleçekimsel Potansiyel enerji (şekil 10) fonksiyonu normal bir fonksiyondur. $E=m*g*h$ ile formülize edilen deneyde kullanıcıdan kütle ve yükseklik alınır ve result içine $m*h$ pointer kullanılarak eklenir.

```
isminizi girin (maksimum 49 karakter): fatih  
Hosgeldin fatih!  
1- Serbest Dusme  
2- Yukari Atis  
3- Agirlik  
4- Potansiyel Enerji  
5- Hidrostatik Basinc  
6- Arsimet Kuvveti  
7- ip Gerilmesi  
8- Basit Sarkac  
9- Asansor  
  
Bir deney seciniz: 4  
Kutlecekimsel potansiyel enerji deneyi secildi  
Kutle(kg): 5  
Yukseklik(m): 15.42  
[ merkur] = 285.27 j  
[ venus] = 683.88 j  
[ dunya] = 756.35 j  
[ mars] = 286.04 j  
[ jupiter] = 1911.31 j  
[ saturn] = 804.92 j  
[ uranus] = 670.00 j  
[ neptun] = 859.66 j
```

Çıktıya baktığımızda(şekil 13); kullanıcı isim girer kullanıcı selamlanır, deney menüsü gösterilir, Potansiyel Enerji deneyi seçilir, cismin kütlesi ve yüksekliği istenir girilen değere göre her gezegen için kütleçekimsel potansiyel enerjisi kullanıcıya gösterilir.

şekil 13

1.2.4 Hidrostatik Basınç

```
void Hidrostatik_Basinc(double *ptr1){  
    /*      P = p*g*h      */  
  
    printf("Hidrostatik basinc deneyi secildi\n");  
    double p,h;  
    while (!guvenli_double_al("Yogunluk(kg/m^3) : ", &p)) printf("Gecersiz!\n");  
    (p<0) ? p*=-1 : (void)0;  
    while (!guvenli_double_al("Derinlik(m) : ", &h)) printf("Gecersiz!\n");  
    (h<0) ? h*=-1 : (void)0;  
  
    *ptr1 += p*h;  
}
```

Şekil 14

Hidrostatik Basınç (Şekil 14) fonksiyonu normal bir fonksiyondur. $P=p*g*h$ ile formüle edilen deneyde kullanıcıdan sıvinin yoğunluk ve derinliği alınır, ardından result içine $p*h$ pointer kullanılarak eklenir.

```
isminizi girin (maksimum 49 karakter): kübra  
Hosgeldin kübra!  
1- Serbest Dusme  
2- Yukari Atis  
3- Agirlilik  
4- Potansiyel Enerji  
5- Hidrostatik Basinc  
6- Arsimet Kuvveti  
7- ip Gerilmesi  
8- Basit Sarkac  
9- Asansor  
  
Bir deney seciniz: 5  
Hidrostatik basinc deneyi secildi  
Yogunluk(kg/m^3): 1200  
Derinlik(m): 6  
[ merkur] = 26640.00 N/m^2  
[ venus] = 63864.00 N/m^2  
[ dunya] = 70632.00 N/m^2  
[ mars] = 26712.00 N/m^2  
[ jupiter] = 178488.01 N/m^2  
[ saturn] = 75168.00 N/m^2  
[ uranus] = 62568.00 N/m^2  
[ neptun] = 80280.00 N/m^2
```

Çıktrıyla baktığımızda (Şekil 15); kullanıcı isim girer kullanıcı selamlanır, deney menüsü gösterilir, Hidrostatik basinc deneyi seçilir, sıvinin yoğunluk ve derinliği alınır ve girilen değere göre her gezegen için sıvinin o derinlikteki hidrostatik basıncını kullanıcıya gösterilir.

Şekil 15

1.2.5 Arşimet Kaldırma Kuvveti

```
void Arsimet_Kaldirma_Kuvveti(double *ptr1) {
    /*      F = p*g*V      */

    printf("Arşimet kaldırma kuvveti deneyi secildi\n");
    double p,V;
    while (!guvenli_double_al("Yogunluk(kg/m^3): ", &p)) printf("Gecersiz!\n");
    (p<0) ? p*=-1 : (void)0;
    while (!guvenli_double_al("Hacim(m^3): ", &V)) printf("Gecersiz!\n");
    (V<0) ? V*=-1 : (void)0;

    *ptr1 += p*V;
}
```

Şekil 16

Arşimet Kaldırma Kuvveti (Şekil 16) fonksiyonu normal bir fonksiyondur. $F=p*g*V$ ile formülize edilen deneyde kullanıcıdan sıvinin yoğunluğu ve sıvı içindeki cismin sıvıda batan hacmi alınır, ardından result içine $p*V$ kullanılarak eklenir.

```
isminizi girin (maksimum 49 karakter): betül
Hosgeldin betül!

1- Serbest Dusme
2- Yukari Atis
3- Agirlik
4- Potansiyel Enerji
5- Hidrostatik Basinc
6- Arsimet Kuvveti
7- ip Gerilmesi
8- Basit Sarkac
9- Asansor

Bir deney seciniz: 6
Arşimet kaldırma kuvveti deneyi secildi
Yogunluk(kg/m^3): 10000
Hacim(m^3): 4
[ merkur] = 148000.00 m^3
[ venus] = 354800.00 m^3
[ dunya] = 392400.02 m^3
[ mars] = 148400.00 m^3
[jupiter] = 991600.04 m^3
[ saturn] = 417599.98 m^3
[ uranus] = 347599.98 m^3
[ neptun] = 445999.98 m^3
```

Çıktıya baktığımızda (Şekil 17); kullanıcı isim girer kullanıcı selamlanır, deney menüsü gösterilir, Arşimet Kuvveti deneyi seçilir, sıvinin yoğunluğu ve sıvı içindeki cismin sıvıda batan hacmi alınır, ardından girilen değere göre her gezegen için kaldırma kuvvetini kullanıcıya gösterilir.

Şekil 17

1.2.6 Sabit İp Gerilmesi

```
void Sabit_ip_Gerilmesi(double *ptr1){  
    /*      T = m.g      */  
  
    printf("Sabit ip gerilmesi deneyi secildi\n");  
    double m;  
    while (!guvenli_double_al("Kutle(kg): ", &m)) printf("Gecersiz!\n");  
    (m<0) ? m*= -1 : (void)0;  
  
    *ptr1 += m;  
}
```

Şekil 18

Sabit İp Gerilmesi (şekil 16) fonksiyonu normal bir fonksiyondur. $T=m.g$ ile formülize edilen deneyde kullanıcıdan cismin ağırlığı alınır ardından result içine doğrudan pointer kullanılarak eklenir.

```
isminizi girin (maksimum 49 karakter): büşra  
Hosgeldin büşra!  
  
1- Serbest Dusme  
2- Yukari Atis  
3- Agirlilik  
4- Potansiyel Enerji  
5- Hidrostatik Basinc  
6- Arsimet Kuvveti  
7- ip Gerilmesi  
8- Basit Sarkac  
9- Asansor  
  
Bir deney seciniz: 7  
Sabit ip gerilmesi deneyi secildi  
Kutle(kg): 6  
[ merkur] = 22.20 N  
[ venus] = 53.22 N  
[ dunya] = 58.86 N  
[ mars] = 22.26 N  
[ jupiter] = 148.74 N  
[ saturn] = 62.64 N  
[ uranus] = 52.14 N  
[ neptun] = 66.90 N
```

Çıktıya baktığımızda(şekil 19); kullanıcı isim girer kullanıcı selamlanır, deney menüsü gösterilir, ip Gerilmesi deneyi secilir, cismin kütlesi alınır altında girilen değere göre her gezegen için ip gerilme kuvveti kullanıcıya gösterilir.

Şekil 19

1.2.6 Basit Sarkaç Periyodu

```
void Basit_Sarkac_Periyodu(const char **isimler,const float *gezegen_g){  
    /*      T = 2*pi*(L/g)^0.5      */  
  
    printf("Basit sarkac periyodu deneyi secildi\n");  
    double L;  
    while (!guvenli_double_al("Ip uzunlugu(m) : ", &L)) printf("Gecersiz!\n");  
    (L<0) ? L=-1 : (void)0;  
  
    for(int i=0;i<8;i++){  
        printf("[%7s] = %8.2lf s\n",*(isimler+i),2*PI*sqrt(L)/sqrt(*gezegen_g+i));  
    }  
}
```

Şekil 20

Basit Sarkaç Periyodu (Şekil 20) fonksiyonu özel bir fonksiyondur. Bu yüzden Gezegen_islem adındaki fonksiyona gitmez, bu yüzden gezegen isim ve ivme değerini pointer olarak parametre alır ve bunları kullanarak doğrudan kullanıcıya çıktı üretir. $T=2\pi(L/g)^{0.5}$ ile formüle edilen deneyde kullanıcıdan sarkacın ip uzunluğu alınır. Çıktı içinse for döngüsü kurulur, her iterasyonda sonraki gezegen ismi, o gezegene ait periyot değerleri yazdırılır.

```
isminizi girin (maksimum 49 karakter): yusufi  
Hosgeldin yusufi!  
1- Serbest Dusme  
2- Yukari Atis  
3- Agirlik  
4- Potansiyel Enerji  
5- Hidrostatik Basinc  
6- Arsimet Kuvveti  
7- ip Gerilmesi  
8- Basit Sarkac  
9- Asansor  
  
Bir deney seciniz: 8  
Basit sarkac periyodu deneyi secildi  
Ip uzunlugu(m): -5  
[ merkur] = 7.30 s  
[ venus] = 4.72 s  
[ dunya] = 4.49 s  
[ mars] = 7.29 s  
[ jupiter] = 2.82 s  
[ saturn] = 4.35 s  
[ uranus] = 4.77 s  
[ neptun] = 4.21 s
```

Şekil 21

Çıktıya baktığımızda (Şekil 21); kullanıcı isim girer kullanıcı selamlanır, deney menüsü gösterilir, Basit Sarkac deneyi seçilir, ipin uzunluğu alınır ardından girilen değere göre her gezegen için ip gerilme kuvveti kullanıcıya gösterilir. Burada ternary operatorünün çalıştığını göstermek için ip uzunluğu negatif girildi ve sonuç pozitif, yani kod düzgün çalışıyor.

1.2.8 Asansör

```
void Asansor(const char** isimler,const float* gezegen_g){
    /* N = m * (g+a) */
    printf("Asansor deneyi secildi\n");
    double m,a;
    while (!guvenli_double_al("Kutle(kg): ", &m)) printf("Gecersiz!\n");
    (m<0) ? m=-1 : (void)0;
    printf("Not:\n");
    printf("## Asansor yukarı yonde ivmelenerek hızlanıyor yada aşağı yonde ivmelenerek yavaşlıyorsa a pozitif gir ##\n");
    printf("## Asansor yukarı yonde ivmelenerek yavaşlıyor yada aşağı yonde ivmelenerek hızlanıyorsa a negatif gir ##\n\n");
    while (!guvenli_double_al("Ivmelenme(m/s^2): ", &a)) printf("Gecersiz!\n");

    for(int i=0;i<8;i++){
        if (*gezegen_g+i) + a <= 0){printf("[%7s] = temas yok\n",*(isimler+i));continue;};
        printf("[%7s] = %8.2lf N\n",*(isimler+i),m * (*gezegen_g+i)+a);
    }
}
```

Şekil 22

Asansör (Şekil 22) fonksiyonu özel bir fonksiyondur. Burada da sarkaç foksiyonunda olduğu gibi gezegen isim ve ivme değerleri pointer ile parametre olarak fonksiyona alınır. Bu fonksiyona özel şöyle bir durum bulunur. $N=m*(g+a)$ ile formülize eğer ki asansör yukarı yönde hızlanıyor veya aşağı yönde yavaşlıyorsa a değeri pozitif, eğer ki asansör yukarı yönde yavaşlıyor veya aşağı yönde hızlanıyorsa a değeri pozitif girilmelidir. Burada normalde duruma göre $(g+a)$ yada $(g-a)$ kullanılması gerekiydi ancak kullanıcının doğrudan negatif pozitif almak daha mantıklıydı. Kullanıcıdan asansör içindeki cismin kütlesi ve bu kurallara göre asansör ivmesi alınır. Daha sonra for döngüsü ile gezegen sayısı kadar döngü döner. Her iterasyonda gezegen ismi ve o gezegene ait sonuç kullanıcıya gönderilir. Eğer ki herhangi bir iterasyonda normal kuvveti negatif çıkıştıysa cisme kuvvet etki etmiyor demektir, bu da döngü içinde if koşulu ile yakalanır ve ona göre çıktı verilir, continue ile döngü devamlılığı sağlanır.

```
Hosgeldin hafız!
1- Serbest Dusme
2- Yukari Atis
3- Agirlik
4- Potansiyel Enerji
5- Hidrostatik Basinc
6- Arsimet Kuvveti
7- ip Gerilmesi
8- Basit Sarkac
9- Asansor

Bir deney seciniz: 9
Asansor deneyi secildi
Kutle(kg): 5
Not:
## Asansor yukarı yonde ivmelenerek hızlanıyor yada aşağı yonde ivmelenerek yavaşlıyorsa a pozitif gir ##
## Asansor yukarı yonde ivmelenerek yavaşlıyor yada aşağı yonde ivmelenerek hızlanıyorsa a negatif gir ##

Ivmelenme(m/s^2): -10
[ merkur] = temas yok
[ venus] = temas yok
[ dunya] = temas yok
[ mars] = temas yok
[jupiter] = 73.95 N
[saturn] = 2.20 N
[ uranus] = temas yok
[ neptun] = 5.75 N
```

Şekil 23

Çıktıya baktığımızda (Şekil 23); kullanıcı isim girer kullanıcı selamlanır, deney menüsü gösterilir, Asansor deneyi seçilir, asansördeki cismin kütlesi ve ivme alınır. Ardından girilen değere göre her gezegen için cisme etki eden normal kuvvet kullanıcıya gösterilir. Yukarıda temassızlık durumuna değişim için negatif ivme girildi, Şekil 23'te görüldüğü gibi bazı gezegenlerde temas yok.

1.3 Girdi Doğrulama ve Hata Yönetimi

Bu kısımda açıkçası oldukça zorluk çektim. Çünkü girilen isimlerin belirli bir karakter sınırını geçmeyecek şekilde yazılması, eğer bu şart sağlanmıyorsa hata mesajı ve tekrar sorma, girilen sayısal değerler integer mi double mi bunların kontrolü, değilse verilecek hata mesajları benim seviyemin çok üzerindeydi. Bu kısımları düz olarak geçebilirdim ancak kodumun gerçekten iyi çalışmasını istiyordum o yüzden bu noktada 3 adet fonksiyon ile bu hata kontrollerini yapmak artık mümkün hale geldi.

```
/*=====NAME__INPUT=====*/
while (1) {
    printf("isminizi girin (maksimum 49 karakter): ");
    if (!fgets(isim, sizeof(isim), stdin)) continue;

    if (!strchr(isim, '\n')) {
        printf("Karakter sınırı aşıldı!\n");
        while (getchar() != '\n');
        continue;
    }

    isim[strcspn(isim, "\n")] = '\0';
    if (isim[0] == '\0') continue;
    break;
}
```

Şekil 24

scanf ile isim almak ilk bakışta kolay gibi görünür ama aslında kontrollsüzdür. String alırken girilen karakter sayısını güvenli biçimde sınırlamaz, boşluk karakterinde okumayı durdurur ve kullanıcı beklenenden uzun bir giriş yaptığında bellek taşmasına yol açabilir. Ayrıca fazla girilen karakterler giriş tamponunda kaldığı için sonraki okumalarda beklenmedik hatalar oluşur. Bu yüzden scanf, özellikle kullanıcının metin alırken güvenilir değildir. Bu kodda fgets kullanmasının temel nedeni, girişi kontrollü ve güvenli biçimde almaktr. fgets, dizinin boyutunu bildiği için buffer taşmasına izin vermez ve satır sonuna kadar olan tüm girişi okur. Böylece kullanıcı boşluk içeren bir isim girse bile tamamı alınır. Kod ayrıca girilen veriyi doğrular. Eğer kullanıcı, dizi boyutunu aşan bir metin girerse satır sonu karakteri (\n) okunmaz. Bu durum strchr ile tespit edilir ve kullanıcıya “karakter sınırı aşıldı” uyarısı verilir. Ardından getchar() döngüsüyle giriş tamponu temizlenir. Bu adım yapılmazsa, fazlalık karakterler bir sonraki girişte sorun çıkarır. Sonrasında strcspn kullanılarak \n karakteri temizlenir ve giriş düzgün bir C string'ine dönüştürülür. Eğer kullanıcı sadece Enter'a basmışsa (yani boş bir isim girdiyse), bu da kontrol edilerek kabul edilmez ve tekrar giriş istenir.

```

int guvenli_int_al(const char *mesaj, int *out)
{
    char buf[64], *end;
    printf("%s", mesaj);

    if (!fgets(buf, sizeof(buf), stdin)) return 0;

    *out = (int) strtol(buf, &end, 10);
    if (end == buf) return 0;

    while (*end == ' ' || *end == '\n') end++;
    return *end == '\0';
}

```

şekil 25

scanf ile tamsayı almak en basit yoldur genelde ama kontrollsüzdür. scanf("%d", &x) başarısız olduğunda giriş tamponunda hatalı veri kalır ve program aynı hatayı tekrar tekrar yaşıar. Ayrıca kullanıcı 123abc gibi bir giriş yaptığından, scanf sayıyı kısmen okur ve geri kalan karakterleri bırakır; bu da sonraki girişleri bozar. Hata durumunu ayırt etmek de zordur, çünkü neden başarısız olduğu net değildir. Bu kodda ise giriş tamamen string olarak alınır. fgets sayesinde okunacak veri tampon boyutuyla sınırlıdır, bu da taşma riskini ortadan kaldırır. Giriş alınamazsa fonksiyon hemen başarısız döner; belirsiz bir durumda ilerlemez. Asıl güvenlik strtol ile sağlanır. strtol, string'i sayıya çevirirken dönüşümün nerede bittiğini end pointer'ı ile bildirir. Eğer end == buf ise, ortada sayı yoktur; bu durum açıkça yakalanır. scanf'de bu ayrımı yapmak çok daha zordur. Sonraki adımda, sayının arkasında sadece boşluk veya satır sonu karakteri olup olmadığı kontrol edilir. Yani 12, 12\n veya 12 geçerli kabul edilirken 12abc kesin olarak reddedilir. scanf ise bu tür hatalı girişleri sessizce kabul edebilir. Bu fonksiyon ile aynı şekilde bir de double için olan versiyonu mevcuttur, aralarındaki fark ise birisi strtol, diğerinde ise strtod kullanılır.

```

isminizi girin (maksimum 49 karakter): aoslklaskdlmaskldmklamkldsmaslkmdkladlsmasdmkadklaskdmksmak
Karakter siniri asildi!
isminizi girin (maksimum 49 karakter): |

```

şekil 26

```

isminizi girin (maksimum 49 karakter): ahmet
Hosgeldin ahmet!
1- Serbest Dusme
2- Yukari Atis
3- Agirlik
4- Potansiyel Enerji
5- Hidrostatik Basinc
6- Arsimet Kuvveti
7- ip Gerilmesi
8- Basit Sarkac
9- Asansor

Bir deney seciniz: 12
Gecerli araligin disinda bir deger girdiniz!

Bir deney seciniz: ali
Gecersiz secim!

Bir deney seciniz: 12ali
Gecersiz secim!

Bir deney seciniz: |

```

şekil 27

```

isminizi girin (maksimum 49 karakter): ali
Hosgeldin ali!
1- Serbest Dusme
2- Yukari Atis
3- Agirlik
4- Potansiyel Enerji
5- Hidrostatik Basinc
6- Arsimet Kuvveti
7- ip Gerilmesi
8- Basit Sarkac
9- Asansor

Bir deney seciniz: 2
Yukari atis deneyi secildi
Ilk hiz(m/s): 23sd
Gecersiz sayi!
Ilk hiz(m/s): 1      asd
Gecersiz sayi!
Ilk hiz(m/s): bir
Gecersiz sayi!

```

şekil 28

Örneklerde(şekil 26, şekil 27

şekil 28) olduğu gibi bu 3 fonksiyon kullanılarak hatalar kullanıcıya gönderilir. Kullanıcı dan isim isterken, deney seçimi yaparken ve deneylerin içinde veri girerken hatalı girişler engellenir bu sayede kod hata- siz bir şekilde çalışır.

2. Eksiklikler ve geliştirmeler

Normal deney fonksiyonlarının her birinin hesapladığı sonucu double türünde doğrudan geri döndürmesi planlanıyordu. Ancak fonksiyonlardan değer döndürme ile pointer üzerinden sonuç aktarma arasındaki farkların tam olarak netleşmesi zaman aldı ve özellikle birden fazla fonksiyonun aynı result değişkeni üzerinde işlem yapması sırasında kafa karışıklığı yaşandı. Bu nedenle daha kontrol edilebilir olduğu düşünülen pointer yaklaşımı tercih edildi. Eğer bu yapı tamamen geri dönüş değerleri üzerine kurulsa, fonksiyonların sorumlulukları daha net ayrılır, yan etkiler azalır ve kod okunabilirliği artardı. Yedi normal deney fonksiyonunda pointer parametrelerin tutarlı ve hatasız biçimde kullanılması hedefleniyordu. Ancak pointer mantığı özellikle başlangıç aşamasında, adres ile değer arasındaki farkın pratikte kavranmasını zorlaştırdı ve yanlış kullanım ihtimali nedeniyle zaman kaybı yaşandı. Bu yüzden bazı fonksiyonlarda daha sade ama tekrarlı çözümler tercih edildi. Pointer kullanımının daha oturmuş bir tasarımla ele alınması, kodun hem esnekliğini hem de profesyonel görünümünü belirgin şekilde artırabilirdi. Hata kontrol mekanizmalarının tüm deney fonksiyonlarında standart hâle getirilmesi amaçlanıyordu. Giriş güvenliği için genel fonksiyonlar yazılmış olsa da, her fiziksel hesap için ayrı ayrı anlamlı hata mesajları ve sınır kontrolleri eklemek zaman açısından mümkün olmadı. Bu noktada özellikle hangi değerin fiziksel olarak anlamsız olduğu konusunda karar vermek zorlayıcı oldu. Daha kapsamlı hata kontrolleri eklenseydi, program kullanıcıyı yalnızca teknik değil, fiziksel açıdan da yönlendiren bir yapıya kavuşurdu. Deney fonksiyonlarının tamamında fiziksel formüllerin eksiksiz biçimde fonksiyon içinde uygulanması düşünülüyordu.

Ancak yerçekimi ivmesinin gezegen bazlı olarak sonradan çarpılması fikri, kod tekrarını azaltmak amacıyla tercih edildi ve bu durum formüllerin parçalanmasına neden oldu. Bu yaklaşım çalışır durumda olsa da kavramsal bütünlüğü zayıflattı. Eğer her fonksiyon kendi formülünü tam olarak uygulasaydı, kod fiziksel anlamda daha sezgisel ve öğretici olurdu. Gezegen sayısı ve gezegen verilerinin tamamen dinamik bir yapıya taşınması planlanıyordu. Fakat sabit dizilerle çalışmak daha hızlı ve anlaşılır olduğu için bu fikir ileriki bir aşamaya bırakıldı. Dinamik bellek ve yapı kullanımı eklenmiş olsaydı, program farklı gezegen setleriyle veya kullanıcı tanımlı verilerle çalışabilecek daha esnek bir hâl alabilirdi. Son olarak geliştirme sürecinde en çok zorlanılan aşamalardan biri, tüm bu fonksiyonel yapıyı bozmadan yeni özellik ekleyebilmek oldu. Özellikle pointer ile veri aktarımı ve güvenli giriş fonksiyonlarının deney fonksiyonlarıyla uyumlu hâle getirilmesi zaman aldı. Bu zorluklar kısa bir zaman aralığında(3 gün) adım adım işlendi önce teoride sonra pratikte çalışması sağlandı.

3. Sonuç

Bu kodun iyi yanlarının başında, C dilinde sıkça ihmal edilen güvenli giriş yaklaşımını merkezine alması geliyor. Benzer seviyedeki birçok muadil programda hâlâ doğrudan scanf kullanılırken, burada girişin önce string olarak alınması ve ardından kontrollü biçimde sayıya dönüştürülmesi, kodu kullanıcı hatalarına karşı çok daha dayanıklı hale getiriyor. Bu fark, programın rastgele bozulmasını veya beklenmedik davranışlar sergilemesini büyük ölçüde engelliyor. Muadillerine kıyasla öne çıkan bir diğer nokta modüler yapı. Deneylerin her birinin ayrı fonksiyonlara bölünmesi, kodu tek parça hâlinde yazılmış uzun main fonksiyonlarından ayıriyor. Bu yaklaşım, hem okuma hem de geliştirme sürecinde ciddi bir avantaj sağlıyor. Yeni bir deney eklemek veya mevcut bir deneyi değiştirmek, programın geri kalanına minimum müdahale gerektiriyor. Kodun dikkat çeken yönlerinden biri de veri ile işlemin ayrılması. Gezegen isimleri, yerçekimi ivmeleri ve birimler sabit dizilerde tutuluyor; hesaplama mantığı bu verilerden bağımsız çalışıyor. Pek çok muadilörnekte bu tür değerler fonksiyonların içine gömülü olurken, burada merkezi ve düzenli bir yapı tercih edilmiş. Bu, hem hata riskini azaltıyor hem de algoritmanın bilimsel tarafını daha net gösteriyor. Ayrıca kullanıcı etkileşimi açısından da muadillerinden ayrılıyor. Geçersiz girişler sessizce geçilmiyor; kullanıcı açıkça uyarılıyor ve doğru giriş yapana kadar yönlendiriliyor. Bu, programı “çalışıyor ama kırılgan” bir örnektен çıkarıp, kullanıcıyla iletişim kurabilen bir uygulama seviyesine yaklaştırıyor. Öne çıkan bir diğer nokta ise algoritmanın öğretici niteliği. Kod sadece sonucu üretmiyor, aynı zamanda C’de güvenli giriş, pointer kullanımını, fonksiyonlar arası veri aktarımı ve dizilerle çalışma gibi temel ama kritik kavramları bir arada ve tutarlı biçimde sergiliyor. Bu açıdan bakıldığından, muadillerine göre daha bilinçli ve mühendislik yaklaşımıyla yazılmış bir çalışma olduğu söylenebilir. Sonuç olarak bu kod; güvenli giriş altyapısı, modüler tasarıımı, veri–işlem ayrimı ve kullanıcı hatalarına karşı dayanıklılığı sayesinde, benzer fizik hesaplama programlarının büyük kısmından daha sağlam, daha okunabilir ve daha geliştirilebilir bir konumda duruyor. Bu da onu sadece çalışan bir örnek değil, aynı zamanda iyi bir referans kod hâline getiriyor.

4. Kaynakça

- Kernighan, B. W., & Ritchie, D. M. (1988). The C programming language (2nd ed.). Prentice Hall.
- Prata, S. (2014). C primer plus (6th ed.). Addison-Wesley.
- ISO/IEC. (2018). ISO/IEC 9899:2018: Programming languages — C. International Organization for Standardization.
- Young, H. D., & Freedman, R. A. (2016). University physics with modern physics (14th ed.). Pearson.
- cppreference.com
- GNU libc manual

Github Linki: https://github.com/mrmumo1212/ALM_LAB_PROJE_MuhammedEminPehlivan_25360859021

