Mağaza Otomasyonu Raporu

Bu projede 15 adet temel fonksiyona sahip bir mağaza otomasyonu gerçekleştirdim. Projenin temeli Tek yönlü bağlı liste veri yapısına dayanıyor. Bu veri yapısı ekle,yazdir,sil,güncelle,araVeYazdir,ara,siraliEkle,urunleriSirala,maliyetHesapla,minBul,maxBul,

urunSayisi ve dosyayaYazdir fonksiyonlarindan oluşmaktadır. Şimdi bu fonksiyonları sırasıyla inceleyelim.

1)ekle Fonksiyonu

```
node* ekle(node* root, int id, string ad, float fiyat) {
    node* temp = new(node);
    temp->id = id;                                  //sondan ürün eklemek için ekle fonksiyonunu oluşturdum
    temp->ad = ad;
    temp->fiyat = fiyat;

    node* iter = root;

    while (iter->next != NULL)
        iter = iter->next;

    iter->next = temp;
    temp->next =NULL;

    return root;
}
```

Bu fonksiyon parametre olarak node* tipinde root bir işaretçi,int türünde bir değişken, string tipinde bir değişken ve float tipindebir değişken almaktadır.

Fonksiyon temp adında yeni bir düğüm oluşturur ve girilen parametreleri bu düğüme atar.

Sonrasında iter işaretçisi ile son elemana kadar listeyi dolaşır ve sona ulaştığında durup sonraki elemana atama işlemi yapar.

2)yazdir Fonksiyonu

```
void yazdir(node* root) {
    while (root!= NULL) {              //tüm ürünleri yazdıran yazdir fonksiyonu
      std::cout << "Urun id: " << root->id << "\tUrun adi: " << root->ad << "\tUrun fiyati: " << root->fiyat << endl;
       root = root->next;
    }
}
```

Bu fonksiyon parametre olarak yalnızca node* tipinde bir işaretçi alır ve bu işaretçi üzerinden tüm listeyi dolaşır. Üzerinden geçtiği her bir elemanı içerikleriyle birlikte ekrana yazdırır.

3)sil Fonksiyonu

```
node* sil(node* root, int id) {

    node* iter = root;
    node* temp;                                  //ürün silmek için sil fonksiyonunu oluşturdum
    if (id == root->id) {
        temp = root;                              //silinen ürünün başta olması bir istisna olduğu için bunu kontrol ettim
        root = root->next;
        free(temp);
    }                                              //silinecek ürün başta değilse genelleştirilmiş silme fonksiyonunu yazdım

    while (iter->next!= NULL && iter->next->id != id) {
        iter = iter->next;
    }
    if (iter->next == NULL) {
        std::cout << "Urun bulunamadi.";
        return root;
    }
    temp = iter->next;
    iter->next = temp->next;
    free(temp);

    return root;

}
```

Bu fonksiyon parametre olarak node* tipinde bir işaretçi ve bir int tipinde değişken alır. Aldığı başlangıç işaretçisinden başlayarak listeyi dolaşır ve istenen id değerinden bir önceki durağa ulaştığında durur ve istenen elemanı siler. Eğer silinecek eleman en başta ise istisna olarak bunu kontrol eder. Eğer silinecek eleman listede mevcut değilse ürünün bulunamadığına dair bir hata mesajı gönderir.

4)güncelle Fonksiyonu

```
node* guncelle(node* root, int id) {
    node* iter = root;
    while (iter != NULL && iter->id != id) {
        iter = iter->next;
    }                                                  //id kontrolu yaparak ürün güncellemesi yapıyorum
    if (iter == NULL) {                                //eğer aranan id ile kayıtlı bir ürün yoksa bir uyarı mesajı gönderiyor.
        std::cout << "Guncellemek istediginiz urun bulunamadi.";
        return root;
    }                                                  //ürün mevcutsa ürünün bulunduğunu belirten bir mesaj ve ürün bilgilerini listeliyor
    std::cout << "Aradiginiz urun bulundu! Mevcut urun bilgileri: " << endl;
    std::cout << "Urun id: " << iter->id << "\tUrun adi: " << iter->ad << "\tUrun fiyati: " << iter->fiyat << endl;

    std::cout << "Urunde yapmak istediginiz guncellemeyi belirleyiniz: " << endl
        << "1-Ad degisikligi" << endl
        << "2-Fiyat degisikligi" << endl                  //kullanıcıya ürün üzeridne nasıl bir değişiklik yapmak istediğini sordum
        << "3-Id degisikligi" << endl;
    int secim;
    cin >> secim;

    switch (secim) {
    case 1:  std::cout << "Ad degisikligi yapmak icin bir ad girin: "; cin >> iter->ad;
        std::cout << "Urun adi basariyla degistirildi!"<<endl;
        break;
    case 2: std::cout << "Fiyat degisikligi yapmak icin bir fiyat girin: "; cin >> iter->fiyat;
        std::cout << "Urun fiyati basariyla degistirildi!"<<endl;
        break;
    case 3: std::cout << "Id degisikligi yapmak icin bir Id girin: "; cin >> iter->id;
        std::cout << "Urun Id basariyla degistirildi!"<<endl;
        break;
    }
    return root;
}
```

Bu fonksiyon node* tipinde bir işaretçi ve int türünde bir değişken parametresi alır.

Aldığı başlangıç işaretçisinden başlayarak aranan id değerini bulana dek listeyi dolaşır. Eğer aranan id değeri listede mevcut değilse bu ürünün bulunmadığına dair bir hata mesajı gönderir.

Eğer ürün bulunursa hangi değişikliğin yapılmak istendiğine dair bir seçim yapılır ve istenen değişiklikler gerçekleştirilir.

5)araVeYazdir Fonksiyonu

```
node* araVeYazdir(node* root) {
    int secim = 0;
    node* iter = root;
    std::cout << "Yapmak istediginiz arama turu: " << endl
        << "1-Id ile arama" << endl
        << "2- Ad ile arama" << endl;
    cin >> secim;
    switch (secim) {
    case 1:
        int id;
        std::cout << "Arama icin id girin: ";
        cin >> id;
        while (iter != NULL && iter->id != id) {
            iter = iter->next;
        }
        if (iter == NULL) {
            std::cout << "Aradiginiz urun bulunamadi";
            return root;
        }

        std::cout << "Aradiginiz urun bulundu! Mevcut urun bilgileri: " << endl;
        std::cout << "Urun id: " << iter->id << "\tUrun adi: " << iter->ad << "\tUrun fiyati: " << iter->fiyat << endl;
        return root;

        break;


    case 2: string ad;
        std::cout << "Arama icin ad girin: ";
        cin >> ad;
        while (iter != NULL && iter->ad != ad) {
            iter = iter->next;
        }
        if (iter == NULL) {
            std::cout << "Aradiginiz urun bulunamadi";
            return root;
        }

        std::cout << "Aradiginiz urun bulundu! Mevcut urun bilgileri: " << endl;
        std::cout << "Urun id: " << iter->id << "\tUrun adi: " << iter->ad << "\tUrun fiyati: " << iter->fiyat << endl;
        return root;

        break;
    }

}
```

Bu fonksiyon parametre olarak yalnızca bir işaretçi alır. Önce id mi yoksa ad ile mi arama yapılacağını belirledikten sonra istenen değer bulunana dek tüm listeyi dolaşır. Aranan ürün bulunursa ürünün detaylarını yazdırır.

Kabaca akış diyagramı

Başla

|

|__ 1. Tek yönü bağlı liste oluştur

| |

| |__ 1.1. İlk ürünü ekle

| |

| |__ 1.1.1. ID, ad ve fiyatı al

| |

| |__ 1.1.2. Yeni ürünü oluştur ve listenin başına ekle

| |

| |__ 1.1.3. "Urun eklendi!" mesajını yazdır

|

|__ 2. Kullanıcıya işlem seçeneği göster

| |

| |__ 2.1. Kullanıcı işlem seçeneği girer

| |

| |__ 2.2. İşlem seçeneğine göre işlem yapılır

| |

| |__ 2.2.1. Urun eklemek için:

| | |

| | |__ 2.2.1.1. ID, ad ve fiyatı al

| | |

| | |__ 2.2.1.2. Yeni ürünü listeye ekle

| | |

| | |__ 2.2.1.3. "Urun eklendi!" mesajını yazdır

| |

| |__ 2.2.2. Urun silmek için:

| | |

| | |__ 2.2.2.1. Silinecek ürünün ID'sini al

| | |

| | |__ 2.2.2.2. Belirtilen ID'ye sahip ürünü listeden sil

| |

| |__ Diğer işlemler için benzer adımlar izlenir

|

|__ 3. Kullanıcı çıkış yapana kadar döngü devam eder

|

Son