Restoran Yemek Takip Projesi (ReYeTaP)

Ders Öğretmeni: **Beste Üstübioğlu**

Grup Numarası: **21. Grup**

Deniz Tunç	434377
Emre Camuzoğlu	439553
Ramazan Karabacak	439705

Özet

Bu çalışmamızda, bir restoranın yemek siparişlerinin, menüsünün ve müşterilerinin takibi için bir Restoran Yemek Takip Projesi (ReYeTaP) ortaya koyduk. Projemizi C programlama dilini kullanarak, optimizasyon ve kod okunurluğunu dengeleyerek yazdık. Projemiz sipariş verme, mevcut ve geçmiş siparişlerin takibi, sipariş onay/ret işlemleri, menü düzenleme, günlük rapor alımı, dönemsel sipariş analizi, ve algoritmik aşçı ataması gibi bir çok özelliğe sahiptir. Projemiz Müşteri, Restoran ve Mutfak olmak üzere 3 ana parçadan oluşmaktadır. Her bir parça aynı anda ve birbiri ile veri alışverişi yaparak çalışmakta ve birbirini tamamlamaktadır.

Projeyi Çalıştırma

Projemiz C programlama dilinde, standartlara çoğunlukla uygun bir biçimde yazılmıştır. Bu sayede projemiz C standart kütüphanesine sahip ve uluslararası standartlara (*ISO-9899*) uygun olduğu sürece işletim sistemi ve derleyici gözetmeksizin çalışabilmektedir. C standartlarında bulunmayıp yazımı işletim sistemine bağlı değişen fonksiyonlar da ön işlemci makroları ile birden fazla platformda çalışabilecek duruma getirilmiştir.

Proje çalışma ortamı ve derleyiciye bağlı olmamakla beraber tavsiye ettiğimiz çalışma ortamı, modülerliği ve kişiselleştirilebilirliği ile ***Visual Studio Code*** adlı metin düzenleyiciden ve standartlara olabildiğince uygun ve hemen hemen her işletim sisteminde çalıştırılabilirliği ile ***GCC*** adlı derleyiciden oluşmaktadır. Projemiz güncel *Microsoft Windows* ve *Linux* platformlarında test edilmiştir.

Müşteri.c

Müşteri uygulaması, restoran müşterilerinin sipariş verebileceği ve verilen siparişleri görebilecekleri arayüzü barındırmaktadır. Ayrıca *müşteri.c*, kod okunurluğu ve modülerlik adına Yemek ve Sipariş girdileri için birer *struct*, bu girdilerin durumlarını belirten birer *enum* ve ilgili dosyalardan girdi alan birer *fonksiyon* içermektedir.

Program başlatıldığında, kullanıcıdan kullanıcı adını girmesini ister. Kullanıcı adı **en fazla 32 karakter** olmak üzere **virgül hariç** boşluk ve özel karakter içerebilmektedir.

```
char kullaniciID[32];
printf("Kullanici Adiniz: ");
scanf("%s", kullaniciID);
printf("Hos geldiniz, %s.\n", kullaniciID);
```

Kullanıcının giriş yapması üzerine ilk karşılaştığı içerik ana menüdür. Burada kullanıcı **sipariş verme (1)** , **siparişlerini görüntüleme (2)** ve **programdan çıkma (0)** arasında seçim yapabilmektedir.

Sipariş Verme

Kullanıcı bu seçeneğe girince *yemek listesi dosyası* okuma, *siparişler dosyası* da sonuna ekleme modunda açılır. Yemek listesi dosyası okunup mevcut olan yemekler kullanıcının seçimi için ekrana yazdırılır. Kullanıcı sipariş etmek istediği yemeği yanında yazılı olan sayı ile seçebilir. Dilerse hiçbir sipariş vermeden çıkmak için de -1 yazarak çıkabilir.

```
int secim;
scanf("%d", &secim);
if (secim == -1) {
printf("Islem iptal edildi.\n");
return;
}
```

Kullanıcının seçtiği yemek, yemek menüsünde durumu “Mevcut Değil” olarak gözüken yemekler atlanarak bulunur ve kullanıcı bilgileri ile beraber *siparişler dosyasına* eklenir.

```
fprintf(siparis_file,
     "%d, %s, %s, %ld, %d, %d, %d\n",
     rand(), kullaniciID, yemek.name,
     time(NULL), yemek.sure, yemek.fiyat, 0);
```

Siparişlerim

Kullanıcının bu seçeneği seçmesi üzerine, *siparişler dosyası* okuma modunda açılır. Dosyadaki siparişler tek tek aktif kullanıcı adı ile karşılaştırılır ve aktif kullanıcıya ait olan siparişler. Ekrana durumları ile yazdırılır. Kullanıcının gün içinde verilmiş siparişi bulunmaması durumunda, bu durum kullanıcıya bildirilir ve kullanıcının giriş yaptığı kullanıcı adını kontrol etmesi önerilir.

```
	int siparis_sayisi = 0;
	while(get_siparis_line(siparis_file, &siparis) == 7){
	if (strcmp(kullaniciID, siparis.user) == 0){
		siparis_sayisi++;

		printf("%s - ", siparis.yemek);

		switch (siparis.durum) {
			case S_BEKLEMEDE:
			printf("Onay Bekliyor.");
			break;
			case S_ONAYLANDI:
			printf("Onaylandi.");
			break;
			case S_HAZIR:
			printf("Hazir.");
			break;
			case S_REDDEDILDI:
			printf("Reddedildi.");
			break;
			default:
			printf("Invalid State");
			break;
		}
		printf("\n");
	}
	}
	if (siparis_sayisi == 0) {
	printf("\tGun icinde verilen siparisiniz
bulunmamaktadir.\n\tKullanici adinizi kontrol ediniz.\n");
	}
```

Restoran.c

Restoran uygulaması, restoran işletmecisinin kullanımı için yapılmış olan arayüzdür. Yüklü miktarda özellik içeren bu uygulama, projemizin en uzun ve en karmaşık parçası olmuştur. Uygulamanın kaynak kodu mutfak uygulamasına benzer şekilde, sipariş ve yemekler için enum ve struct yapılarına sahiptir.

Uygulamanın ana menüsü *yemek listesi güncelleme* (1), *sipariş onay/ret* (2), *rapor işlemleri* (3), *analiz işlemleri* (4) ve *çıkış* (-1) seçeneklerinden oluşmaktadır. Kullanıcı, müşteri uygulamasında olduğu gibi dilediği seçeneğin yanındaki sayıyı girerek, ilgili işlemi yapabilmektedir.

Yemek Listesi Güncelleme

Güncelleme menüsü, kullanıcıya sunulan yemek listesi ögelerinin eklenmesi, silinmesi ve düzenlenmesi için kullanılmaktadır. Kullanıcı, dilerse bu üç seçenekten birini seçebilir, dilerse ana menüye dönmek için girdiye -1 yazabilir.

Yemeklerin eklenmesi:

Yemek dosyası ekleme modunda açılır ve kullanıcıya eklenecek yemeğin bilgileri sorulur. Kullanıcı eklenecek yemeğin *ismini, fiyatını, hazırlanma süresini* ve *mevcudiyetini* girmelidir. Verilen bilgiler, dosyanın sonuna eklenerek kaydedilir. İşlem sonunda yemek listesinin yeni hali ekrana yazdırılır.

```
      Yemek yemek;
      printf("Eklenecek yemek adi giriniz : ");
      scanf("\n%[^\n]", yemek.ismi);

      printf("Eklenecek yemek fiyati girin : ");
      scanf("%d", &yemek.fiyati);

      printf("Eklenecek yemek hazirlanma suresi girin : ");
      scanf("%d", &yemek.suresi);

      printf("Eklenecek yemek durumu girin (0 veya 1 | 0 =
Mevvut degil, 1 = Mevcut):");
      scanf("%d", &yemek.durumu);

      fprintf(yemeklistesi_ekle, "%s, %d, %d, %d\n",
              yemek.ismi, yemek.fiyati, yemek.suresi,
              yemek.durumu);
```

Yemeklerin silinmesi:

Yemek dosyası okuma modunda açılır, arabellek olarak kullanılacak kopya dosyası yazma modunda açılır. Yemek dosyasının içerikleri kopya dosyasına aktarılır. Kopya dosyasına veri aktarılmadıysa yemek listesi boş

olduğundan işlem sonlandırılır. Aktarım sonunda yemek listesi ileride bir daha yazma modunda açılmak üzere kapatılmaktadır.

Bu işlemler sonunda geçerli yemek listesi ekrana yazdırılır ve kullanıcıya silinmesi istenen satır sorulur. Silinecek satır haricindeki satırlar yazma modunda yeniden açılan yemek listesi dosyasına yazdırılır, dosyalar kapatılır, ve kopya dosya silinir.

```
      printf("Silmek istediginiz yemegin bulundugu satiri
giriniz : ");
      scanf("%d", &l);
      ...
      counter = 0;
      while(fgets(kelime_tutucu, 64, yemeklistesi_sil_kopya)){
            counter++;
            if(l == counter) continue;
            fprintf(yemeklistesi_yeni, "%s", kelime_tutucu);
      }
```

Yemek bilgilerinin düzenlenmesi:

Yemek dosyası okuma modunda açılır, arabellek olarak kullanılacak kopya dosyası yazma modunda açılır. Yemek listesi ekrana yazdırılır. Kullanıcıya düzenlenecek satır sorulur. Düzenlenecek satır dışındaki satırlar kopya dosyaya yazdırılır, düzenlenecek satıra gelince kullanıcıya yemeğin özellikleri sorulur ve kopya dosyaya yeni yemek yazdırılır.

```
while ((c = fgetc(yemeklistesi_duzenle)) != EOF){
      fseek(yemeklistesi_duzenle, -1, SEEK_CUR);
      //duzenlencek satir
      if(sayac == l){
            fscanf(yemeklistesi_duzenle, "%[^,], %d, %d, %d\n",
                  duzenle.ismi, &duzenle.fiyati,
                  &duzenle.suresi,&duzenle.durumu);
            ..............
            fprintf(yemeklistesi_duzenle_kopya, "%s, %d, %d,%d\n",
                  duzenle.ismi, duzenle.fiyati,
                  duzenle.suresi, duzenle.durumu);
            sayac++;
            continue;
      }
      //kopyalama
      fgets(kelime_tutucu, 64, yemeklistesi_duzenle);
      fprintf(yemeklistesi_duzenle_kopya, "%s", kelime_tutucu);
```

```
            sayac++;
    }
```

Özellikleri düzenlerken bilgileri_duzenle fonksiyonu kullanılır. Fonksiyon, aldığı girdi -1 ‘e eşit ise orijinal değeri, değilse girdiden elde ettiği değeri döndürmektedir.

```
int bilgileri_duzenle(int x){
      int r;
      scanf("%d", &r);
      //-1 degilse girilen degeri dondur
      if(r != -1)
      return r;
      return x;
}
```

İşlem sonunda yeni liste ekrana yazdırılır ve dosyalar kapatılır.

Sipariş Onay/Ret

Müşterilerin verdiği siparişler mutfakta hazırlanmaya başlamadan önce restoran tarafından onaylanmalıdır. Sipariş dosyası okuma, kopya dosyası yazma modunda açılır. Sipariş durumu *“Onay Bekliyor” (0)* olan siparişler ekrana yazdırılır. Onaylanacak satır bulunmaması durumunda, durum kullanıcıya bildirilir ve işlem sonlandırılır.

```
     while (fscanf(siparis_dosya, "%d, %[^,], %[^,], %ld, %d, %d,
%d\n", &id, kullanici, yemek, &zaman, &sure, &fiyat, &durum) > 0){

     // Eger degistirilecek satiri daha yazdirmadiysak
     if(!done){
           if (durum == 0) counter++;
           if (counter == change_line) {
                 // Yeni durumu kullanicidan iste
                 done = 1;
                 printf("Onay (1), Ret (-1) %d: ", id);
                 scanf("%d", &durum);
                 if (durum == -1) durum = S_REDDEDILDI;
           }
     }

     fprintf(kopya_dosya, "%d, %s, %s, %ld, %d, %d, %d\n",
           id, kullanici, yemek, zaman, sure, fiyat, durum);
```

```
}
```

Onaylanacak satır kullanıcıya sorulur. Seçilen satır dışındaki satırlar aynı şekilde kopya dosyaya aktarılır, seçilen satıra gelince onay durumu sorulur ve satır yeni hali ile dosyaya yazılır. İşlem sonunda kopya dosyanın içeriği asıl dosyaya aktarılır ve dosyalar kapatılır.

Rapor İşlemleri

Siparişler dosyasının aşırı büyümesini engellemek için gün sonunda hazırlanan sipariş raporları arşive gönderilmelidir. Kullanıcı bu menüde günlük raporları oluşturabilir ve görüntüleyebilir.

Rapor oluşturma seçildiğinde geçerli tarihe göre bir rapor dosyası oluşturulur ve sipariş durumu "Hazır" veya "Reddedildi" olan siparişler rapor dosyasına aktarılır.

Rapor görüntüleme seçildiğinde, görüntülenmek istenen tarih gg.aa.yy şekline kullanıcıdan alınır. Verilen tarihe göre uygun dosya açılır ve dosyada bulunan siparişler ekrana yazdırılır.

```
scanf("%d.%d.%d", &tarih[0], &tarih[1], &tarih[2]);
sprintf(dosya_ismi, "arsiv/%02d.%02d.%02d.txt",
     tarih[0], tarih[1], tarih[2]);
FILE* dosya = fopen(dosya_ismi, "r");
```

Analiz İşlemleri

Restoranın istatistiklerini görüntülemek için Analiz İşlemleri menüsü kullanılır. Analiz işlemleri günlük kazanç, aylık kazanç, dönemsel kazanç ve en çok tüketim olmak üzere dörde ayrılır.

Günlük/Aylık/Dönemsel Kazanç:

İncelenmesi istenen tarih veya tarih aralığı kullanıcı tarafından girilir. Girilen tarihe göre ilgili arşiv dosyaları okuma modunda açılır. Dosya içindeki durumu "Hazır" olan siparişlerin fiyat bilgisi toplanır ve ekrana yazdırılır.

En çok tüketim:

En çok tüketim seçeneği, restoranın arşivindeki bütün dosyaları incelemeye alır. İncelenecek istatistikler için birer struct açılır (Yemek, Kullanıcı, Gün). Klasörde bulunan her hazırlanmış sipariş için istatistik listelerinin ilgili ögelerinin sayacı bir arttırılır.

İstatistikler sayılınca, yazdırılmak üzere, önceden tanımlanmış bubble sort fonksiyonuna gönderilir. Burada istatistikler büyükten küçüğe sıralanır ve

dosyanın en üstüne önişlemci makrosu olarak tanımlanan TOP_RANK_SIZE değişkenine göre ilk sırada olanlar ekrana yazdırılır.

```
bubble_sort(yemekler, yemekler_len, sizeof(yemek));
bubble_sort(users, users_len, sizeof(user));
bubble_sort(gunler, gunler_len, sizeof(gun));

printf("\n-> En cok siparis veren kullanicilar:\n");
for (int i = 0; i < TOP_RANK_SIZE; i++) {
     if (i >= users_len) {
          printf("%d#                -\n", i + 1);
     }
     else printf("%d# %4d siparis ile - %s\n",
          i + 1,
          users[i].count,
          users[i].username);
}
printf("\n-> En cok siparis verilen yemekler:\n");
for (int i = 0; i < TOP_RANK_SIZE; i++) {
     if (i >= yemekler_len) {
          printf("%d#                -\n", i + 1);
     }
     else printf("%d# %4d siparis ile - %s\n",
          i + 1,
          yemekler[i].count,
          yemekler[i].name);
}
printf("\n-> En cok siparis verilen gunler:\n");
for (int i = 0; i < TOP_RANK_SIZE; i++) {
     int* tarih = gunler[i].tarih;
     if (i >= gunler_len) {
          printf("%d#                -\n", i + 1);
     }
     else printf("%d# %4d siparis ile - %d.%d.%d\n",
          i + 1,
          gunler[i].count,
          tarih[0], tarih[1], tarih[2]);
}
printf("\n-> Ascilarin toplam calisma suresi: %lddk",
          toplam_hazirlama_suresi);
```

Mutfak.c

Mutfak.c, siparişleri hazırlayan aşçıların olduğu programdır. Mutfak programı, kullanıcıdan aktif girdi gerektirmez, kendi başına sonsuz bir döngü içinde çalışır. Sipariş dosyasındaki onaylanmış siparişler taranıp, sıraya konulur. Boşta aşçı bulunması durumunda sipariş sırasındaki ilk yetiştirilmesi gerek sipariş tespit edilir ve aşçı listesindeki boş bir aşçıya atanır. Çalışan aşçı sayısı, dosyanın en üstünde bulunan bir makro ile ayarlanabilir.

```
      // Asciya yeni gorevini ver
      int idx = priority_job_index(current_time);
      if (idx == -1) {
            bosta_asci_var = 1;
            asci->bitis_zamani = 0;
            asci->siparis.id = 0;
            continue;
      }
      Siparis s = queue[idx];
      ascilar[a].siparis = s;
      ascilar[a].bitis_zamani = current_time + s.sure * SURE_CARP;
      pop_queue(idx);
```

Aşçı uyku optimizasyonu makrosunun 1 olması ve tüm aşçıların dolu olması durumunda aşçılardan bitiş zamanı en yakın olan aşçının kalan süresi kadar program uyku moduna geçirilir.

```
     if (!bosta_asci_var && ASCI_SLEEP_OPTIMIZASYON) {
           int ilk_bitiren = 0;
           for (int a = 0; a < ASCI_SAYISI; a++) {
                 if (ascilar[a].bitis_zamani <
ascilar[ilk_bitiren].bitis_zamani) {
                 ilk_bitiren = a;
                 }
           }
           DEBUG(
                 printf("Tum ascilar dolu, %ld saniye uykuya geciliyor!\n",
(ascilar[ilk_bitiren].bitis_zamani - current_time));
                 fflush(stdout);
           )
           SLEEP((ascilar[ilk_bitiren].bitis_zamani - current_time));
     }
     else SLEEP(1);
```

Herhangi bir aşçının işi bittiğinde “*durum_güncelle*” fonksiyonu çağırılır ve ilgili aşçıya atanan siparişin durumu, *Hazır (2)* olarak güncellenir.

```
      while (fgetc(siparisler_file) != EOF){
      fseek(siparisler_file, -1, SEEK_CUR);
      fscanf(siparisler_file, "%d, %[^,], %[^,], %ld, %d, %d,
%d\n", &cid, user, dish, &zaman, &sure, &fiyat, &cdurum);

      if(cid == id) fprintf(copy_file, "%d, %s, %s, %ld, %d, %d,
%d\n", id, user, dish, zaman, sure, fiyat, durum);
      else fprintf(copy_file, "%d, %s, %s, %ld, %d, %d, %d\n", cid,
user, dish, zaman, sure, fiyat, cdurum);
      }
```

Programın aşırı derecede işlem yapmasını engellemek için sonsuz döngünün içinde 1 saniyelik uyku modu kullanılır.