İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

FILE ORGANIZATION HOMEWORK 3

MUHAMMET TALHA ODABASI
1306220012

1. LINKED LIST

Şekillerde görüldüğü üzere C++ üzerinde linkedList class yapısı oluşturdum ve düğüm yazdırma, ekleme, silme gibi fonksiyonlar ekledim.

```
class linkedList{
    private:
        struct node{
            student data;
            node* next;
        };
        node* head;
        node* tail;
    public:
        linkedList(){
            head = NULL;
            tail = NULL;
        }
        void addNode(student data){
            node* temp = new node;
            temp->data = data;
            temp->next = NULL;
            if(head == NULL){
                head = temp;
                tail = temp;
            }else{
                tail->next = temp;
                tail = tail->next;
            }
        }

        void printNodes(){
            node* temp = head;
            while(temp != NULL){
                printf("Ogrenci No: %d\n", temp->data.ogr_no);
                printf("Ogrenci Ad: %s\n", temp->data.ad);
                printf("Ogrenci Soyad: %s\n", temp->data.soyad);
                printf("Ogrenci BOlum: %s\n", temp->data.bolum);
                printf("\n");
                temp = temp->next;
            }
        }

        void deleteNode(student data){
            node* temp = head;
            node* prev = NULL;
            while(temp != NULL){
                if(temp->data.ogr_no == data.ogr_no){
                    if(prev == NULL){
                        head = temp->next;
                        delete temp;
                        return;
                    }else{
                        prev->next = temp->next;
                        delete temp;
                        return;
                    }
                }
                prev = temp;
                temp = temp->next;
            }
        }
};
```

```
struct Student{
    int ogr_no;
    char ad[20];
    char soyad[20];
    char bolum[40];

} typedef student;
```

main içeriği şekilde görüldüğü gibidir. “students.bin” dosyamı güncelleme modunda açıp ilk halini linkedList içerisine atıyorum ve kullanıcının seçimine göre işlemler yapıyorum.

```
int main(){
    linkedList *studentList = new linkedList();
    FILE *students;
    student blank = {-1, "", "", ""};

    students = fopen("students.bin", "rb+");
    if (!students)
        return 0;
    else{
        rewind(students);
        fread(&blank, sizeof(student), 1, students);
        while (!feof(students)){
            studentList->addNode(blank);
            fread(&blank, sizeof(student), 1, students);
        }
        rewind(students);

        int choise = getchoise();

        switch (choise){
            case 1:
                ogrenciEkle_1(students, blank);
                studentList->addNode(blank);
                break;
            case 2:
                ogrenciSil_2(students, blank);
                studentList->deleteNode(blank);
                break;
            case 3:
                ogrenciGuncelle_3(students, blank, studentList);
                studentList->addNode(blank);
                break;
        }
    }
    printf("\n\n");
    studentList->printNodes();
    fclose(students);
}
```

```
int getchoise(){
    int a;

    printf("[1] Ogrenci Ekle\n"
        "[2] Ogrenci Sil\n"
        "[3] Ogrenci Guncelle\n"
        "Seciminiz >> ");
    scanf("%d", &a);
    while (!(a >= 1 && a <= 3))
    {
        fflush(stdout);
        printf("1-4 arasi secim giriniz >> ");
        scanf("%d", &a);
    }
    return (a);
}
```

Kullanıcı ilk olarak şekildeki menu içerisinden seçimini yapıyor. Seçim main' e returnedilip switch case ile fonksiyon çağırıyolar.

```
void ogrenciEkle_1(FILE *f, student& blank){
    fflush(stdout);
    printf("Ogrenci No: ");
    scanf("%d", &(blank.ogr_no));
    fflush(stdout);
    printf("Ogrenci Ad: ");
    scanf("%s", blank.ad);
    fflush(stdout);
    printf("Ogrenci Soyad: ");
    scanf("%s", &(blank.soyad));
    fflush(stdout);
    printf("Ogrenci BOlum: ");
    scanf("%s", blank.bolum);

    fseek(f, 0, SEEK_END);
    fwrite(&blank, sizeof(student), 1, f);
}
```

Öğrenci ekleme fonksiyonu şekilde görüldüğü gibi gerekli girdileri alıp fwrite ile dosya sonuna yeni öğrenciyi ekliyor. Bu fonksiyon çağrıldıktan sonra case içerisinde linkedList addNode fonksiyonunu çağırıyor ve yeni öğrenciyi düğüme ekliyor.

```
void ogrenciSil_2(FILE *f, student& blank){
    int a;

    printf("Ogrenci No >> ");
    scanf("%d", &a);

    FILE *tmp_f = fopen("tmp.bin", "wb");
    fread(&blank, sizeof(student), 1, f);
    while(!feof(f)){
        if (blank.ogr_no != a){
            fwrite(&blank, sizeof(student), 1, tmp_f);
        }
        fread(&blank, sizeof(student), 1, f);
    }
    fclose(tmp_f);
    remove("students.bin");
    rename("tmp.bin", "students.bin");
}
```

Öğrenci silme işlemi ise öğrenci no' su verilen öğrenci bulunup yeni dosyaya o kayıt olmadan yazılıyor.
Fonksiyon çağrıldıktan sonra case içerisinde deleteNode fonksiyonu çağrılıp düğüm linkedList içerisinden kaldırılıyor.

```
void ogrenciGuncelle_3(FILE *f, student &blank, linkedList *studentList){
    int a;
    int index;

    printf("Ogrenci No >> ");
    scanf("%d", &a);

    index = 0;
    fseek(f, 0, SEEK_SET);
    fread(&blank, sizeof(student), 1, f);
    while (!feof(f)){
        if (blank.ogr_no == a)
            break;
        fread(&blank, sizeof(student), 1, f);
        index++;
    }
    if (blank.ogr_no != a){
        printf("ogrenci bulunamadi"); return;
    }
    studentList->deleteNode(blank);
    fflush(stdout);
    printf("Ogrenci Ad: ");
    scanf("%s", blank.ad);
    fflush(stdout);
    printf("Ogrenci Soyad: ");
    scanf("%s", &(blank.soyad));
    fflush(stdout);
    printf("Ogrenci BOlum: ");
    scanf("%s", blank.bolum);
    fseek(f, sizeof(student) * index, SEEK_SET);
    fwrite(&blank, sizeof(student), 1, f);
}
```

Son olarak öğrenci güncelleme için no' su verilen öğrenci bulunup üzerine kayıt yazılıyor.
Fonksiyon çağrıldıktan sonra önce nosu verilen öğrenci düğümü bulunup linkedList içerisinden kaldırılıyor sonra sona tekrar ekleniyor. Böylece linkedList güncellik sıralamasına göre sıralı olmuş oluyor.

Tüm işlemler sonunda linkedList' in çalıştığını gösterebilmek için printNodes fonksiyonu çağrılıyor ve tüm düğümler yazdırılıyor.

2. HASHTABLE

Yan tarafta örnek hashtable gösterilmiştir. HashTable aslında indexlenmiş LinkedList veya LinkedList arrayi olarak değerlendirilebilir. Sözlük tarzı yapılarda kullanılmaktadır.

HashTable içerisinde indexlemeyi öğrenci no son 2 haneye göre yaptım yani 1306220012 ve 1306210012 aynı index altındaki LinkedList içerisinde bulunacaklar.

Fonksiyonlar ve işlevleri ise LinkedList yapısı ile aynıdır.

```
class hashTable{
    public:
        hashTable(){
            for (int i = 0; i < 100; i++)
                table[i] = NULL;
        }
        void addNode(student &s){
            int index = s.ogr_no % 100;
            if (!table[index]){
                table[index] = new node(s);
            }
            else{
                node *tmp = table[index];
                while (tmp->next)
                    tmp = tmp->next;
                tmp->next = new node(s);
            }
        }
        void deleteNode(student &s){
            int index = s.ogr_no % 100;
            if (!table[index])
                return;
            node *tmp = table[index];
            if (tmp->data.ogr_no == s.ogr_no){
                table[index] = tmp->next;
                delete tmp;
                return;
            }
            while (tmp->next){
                if (tmp->next->data.ogr_no == s.ogr_no){
                    node *tmp2 = tmp->next;
                    tmp->next = tmp->next->next;
                    delete tmp2;
                    return;
                }
                tmp = tmp->next;
            }
        }
        void printNodes(){
            for (int i = 0; i < 100; i++){
                node *tmp = table[i];
                while (tmp){
                    printf("\nOgrenci No: %d\n", tmp->data.ogr_no);
                    printf("Ogrenci Ad: %s\n", tmp->data.ad);
                    printf("Ogrenci Soyad: %s\n", tmp->data.soyad);
                    printf("Ogrenci Bolum: %s\n", tmp->data.bolum);
                    tmp = tmp->next;
                }
            }
        }
    private:
        struct node{
            student data;
            node *next;
            node(student &s){
                data = s;
                next = NULL;
            }
        } *table[100];
};
```

```
int main(){
    FILE *students;
    hashTable *studentTable = new hashTable();
    student blank = {-1, "", "", ""};

    students = fopen("students.bin", "rb+");
    if (!students)
        return 0;
    else{
        rewind(students);
        fread(&blank, sizeof(student), 1, students);
        while (!feof(students)){
            studentTable->addNode(blank);
            fread(&blank, sizeof(student), 1, students);
        }
        rewind(students);

        int choise = getchoise();

        switch (choise){
            case 1:
                ogrenciEkle_1(students, blank);
                studentTable->addNode(blank);
                break;
            case 2:
                ogrenciSil_2(students, blank);
                studentTable->deleteNode(blank);
                break;
            case 3:
                ogrenciGuncelle_3(students, blank, studentTable);
                studentTable->addNode(blank);
                break;
        }
    }
    printf("\n\n");
    studentTable->printNodes();
    fclose(students);
}
```

main fonksiyonu dışında herhangi bir değişklik yapılmamıştır LinkedList' e nazaran. Şekilde görüldüğü üzere hashtable içerisine başlangıçta dosya içerisinden okunarak tüm öğrenci kayıtları ekleniyor. Ekleme sonrasında kullanıcının seçimine göre hep dosya hem de hashtable için gerekli işlemler yapılıyor. En sonda ise hashtable yapısının çalıştığını göstermek için hashtable' ın tüm indexlerindeki LinkedList yapıları sıra ile listeleniyor.