CME 112- Programming Languages II

Week 9

Singly Linked Linear Lists

Assist. Prof. Dr. Caner Özcan

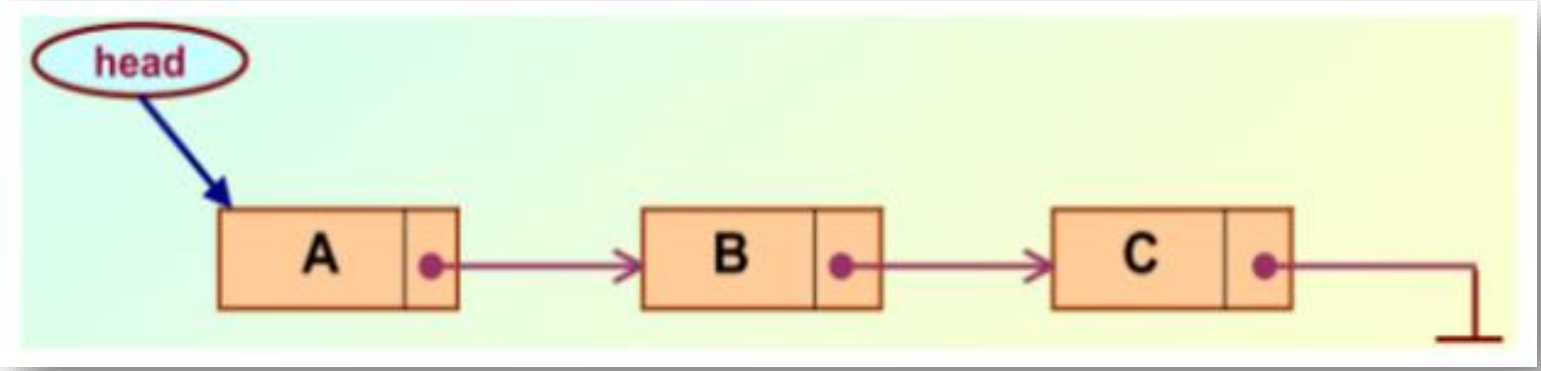
Understanding is the beginning of love. B. Spinoza

Linked Lists

- Linked lists are useful to study for some reasons.
- Most obviously, linked lists are a data structure for real programming.
- Knowing the strengths and weaknesses of linked lists will help you thinking about complexity of processing time and memory space of algorithms.
- Linked lists are a good way of understanding the pointers.
- Linked list problems are a nice combination of algorithms and pointer manipulation.

Linked Lists

- A linked list is a data structure that can be changed during execution.
- Consecutive elements are connected with the pointer.
- Last element points to NULL.
- Size can grow or shrink during the execution of the program (It can be made just as long as required)
- It doesn't made waste memory

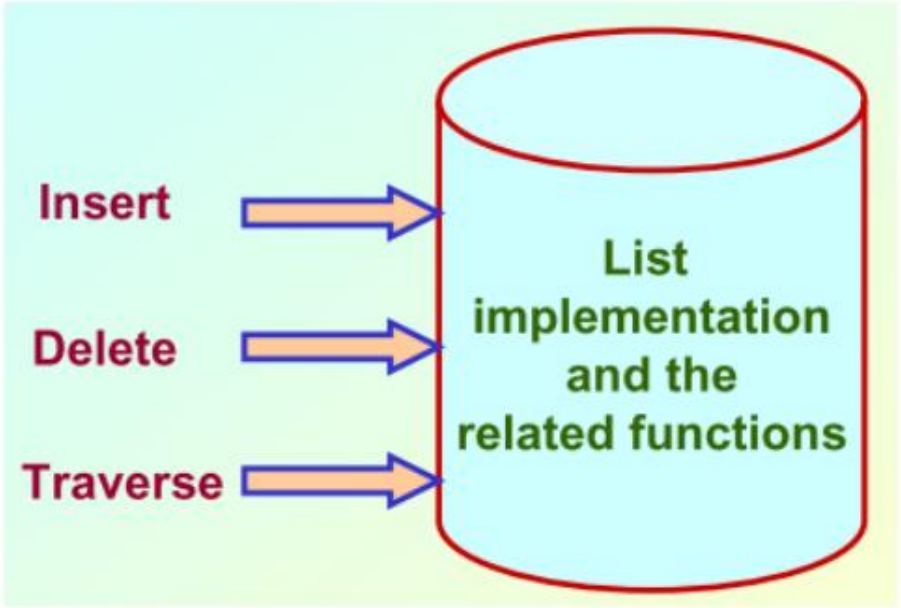


Arrays vs. Linked Lists

- Arrays are suitable for:
  - Inserting/deleting an element at the end.
  - Randomly accessing any element.
  - Searching the array for a particular value.
- Linked lists are suitable for:
  - Inserting an element.
  - Deleting an element.
  - Applications where sequential access is required.
  - In situations where the number of elements can not be predicted beforehand.

Linked Lists

- List is an abstract data type
- This data type is defined by the user.
- Typically more complex than simple data types like int, float, etc.
- Main aim is;



Basic Operations on a List

- Creating a list
- Traversing the list
- Inserting an item to the list
- Deleting an item from the list
- Concatenating two lists into one

Linked Lists

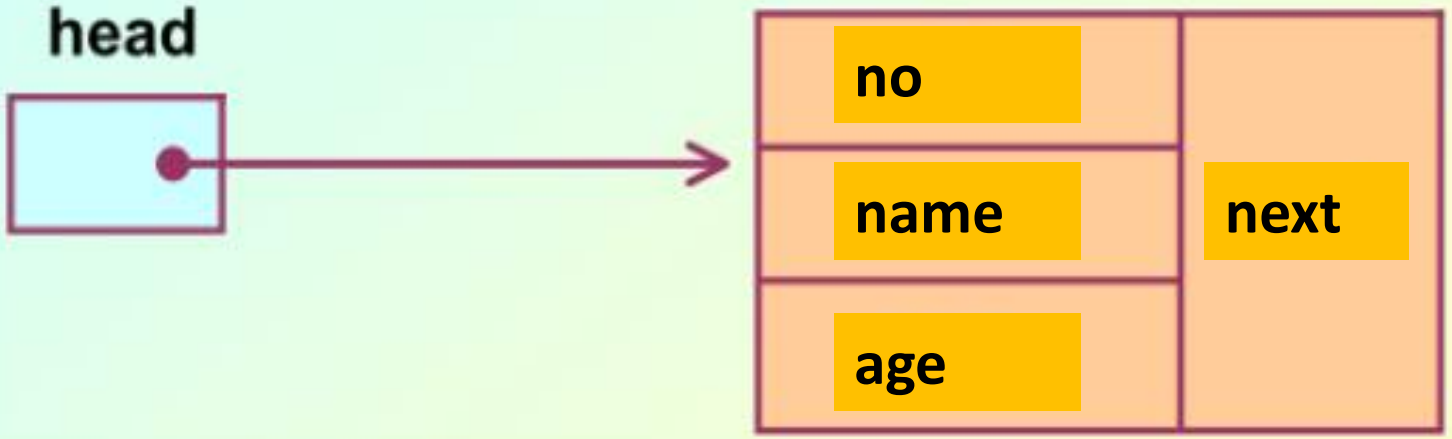
- Consider the structure of a node in the list as follows:

```
#include <stdio.h>
#include <stdlib.h>

struct student{
    int no;
    char name[40];
    int age;
    struct student *next;
};
typedef struct student node;
node *head,*newNode;
```

Creating a Linear List

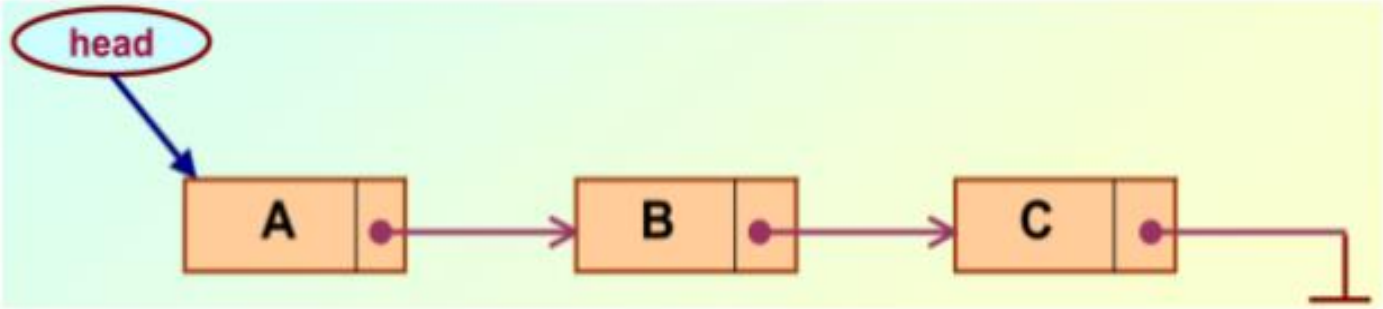
- First, a node must be created and the head must be provided to point that node.

```
head = (node *) malloc(sizeof(node));
```



Creating a Linear List

- If number of nodes is n in the initial linked list:
  - Allocate n records, one by one.
  - Read in the fields of the records.
  - Modify the links of the records so that the chain is formed.



Creating a Linear List


```
node* createList(){
    int n,k;
    node *head,*p;
    printf("How many students in the list?");
    scanf("%d",&n);
    for(k=0;k<n;k++){
        if(k==0){
            head = (node *)malloc(sizeof(node));
            p = head;
        }
        else{
            p->next = (node *) malloc(sizeof(node));
            p = p->next;
        }
        printf("Enter %d. student number: ",k+1); scanf("%d",&p->no);
        printf("Enter %d. student name: ",k+1); scanf("%s",p->name);
        printf("Enter %d. student age: ",k+1); scanf("%d",&p->age);
    }
    p->next = NULL;
    return head;
}
```

Traversing the List

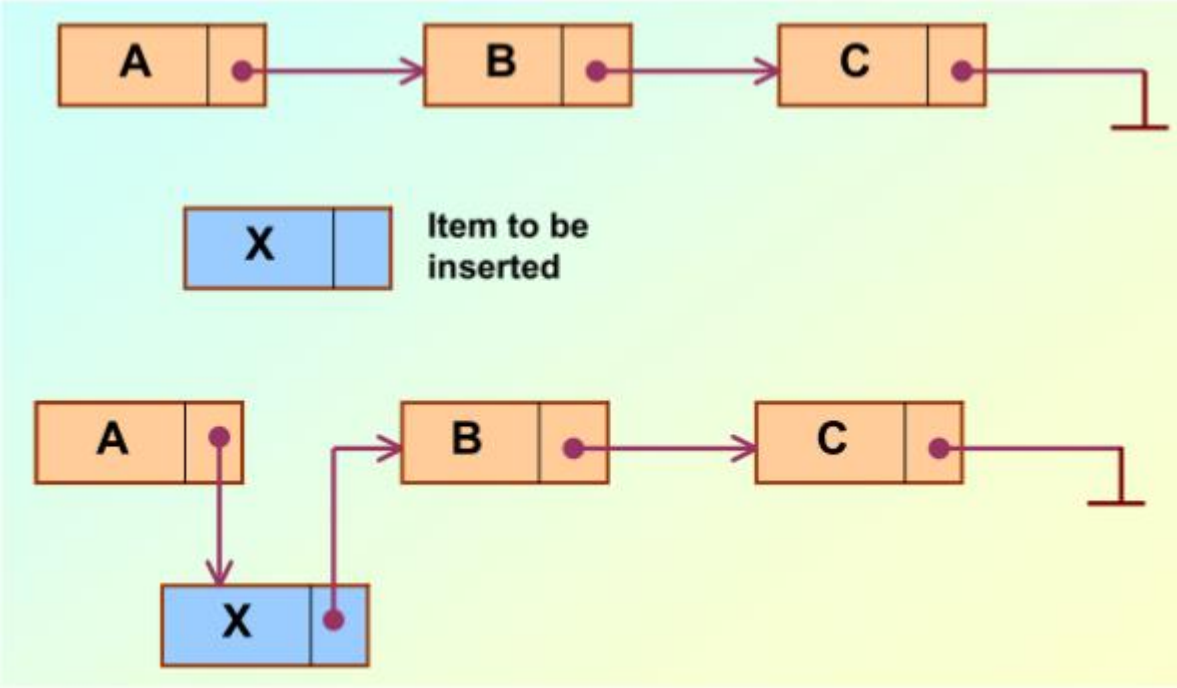
- Once the linked list has been constructed and head points to the first node of the list:
  - Follow the pointers.
  - Display the contents of the nodes as they are traversed.
  - Stop when the next pointer points to NULL

```
void traverseList(node* head){
    int counter =1;
    node *p;
    p = head;
    while(p!=NULL){
        printf("%d- %d %s %d \n",counter,p->no,p->name,p->age);
        p = p->next;
        counter++;
    }
}
```

Adding a Node to a List

- For insertion:
  - A record is created holding the new item.
  - The next pointer of the new record is set to link it to the item which is to follow it in the list.
  - The next pointer of the item which is to precede it must be modified to point to the new item.
- The problem is to insert a node before a specified node.
  - Specified means some value is given for the node (called key).
  - In this example, we consider it to be number.

Adding a Node to a List



Adding a Node to a List

- When a node is added to the beginning,
  - Only one «next» pointer needs to be modified.
    - Head is made to point to the new node.
    - New node points to the previously first element.
- When a node is added to the end,
  - Two «next» pointers need to be modified.
    - Last node now points to the new node.
    - New node points to NULL
- When a node is added to the middle,
  - Two «next» pointers need to be modified.
    - Previous node now points to the new node.
    - New node points to the next node.

Adding a Node to a List

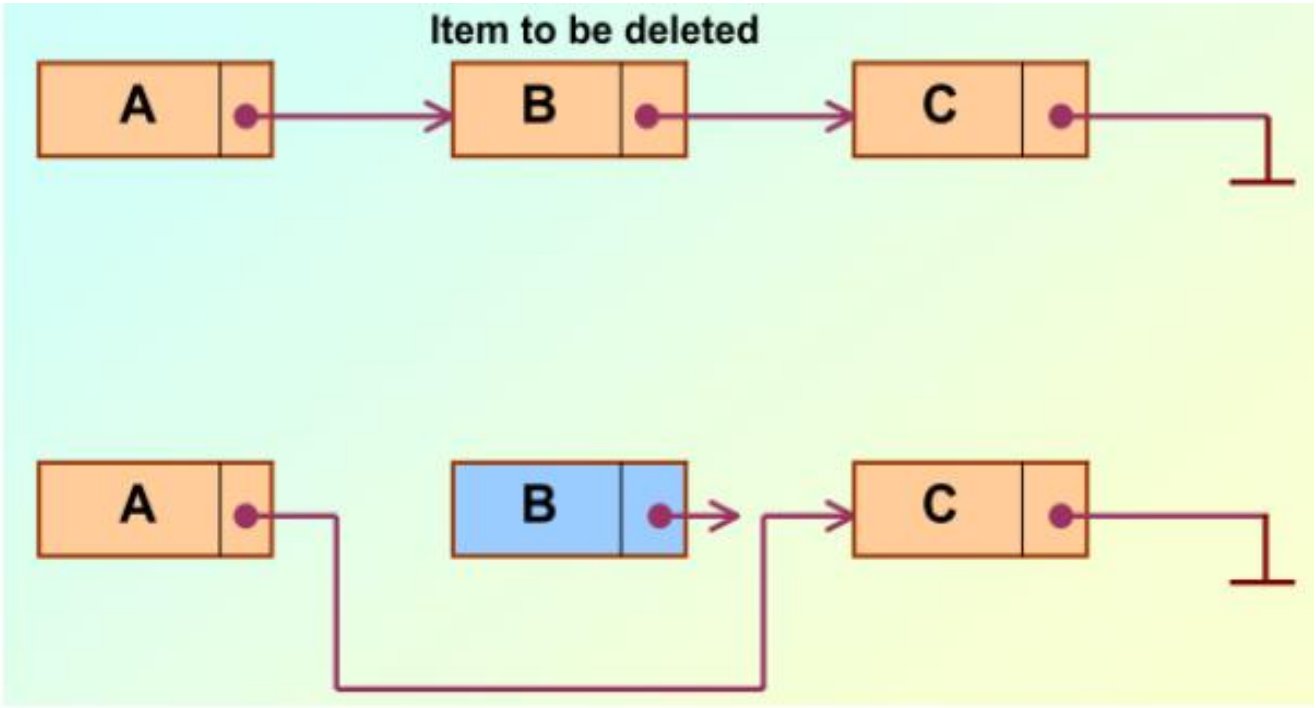

```
node* addNode(node* head){
    int stdNo;
    node *p, *q;
    node *newNode = (node *) malloc(sizeof(node));
    printf("Enter new student number: "); scanf("%d",&newNode->no);
    printf("Enter new student name: "); scanf("%s",newNode->name);
    printf("Enter new student age: "); scanf("%d",&newNode->age);

    printf("Enter std number that new record will be added before: \n");
    printf("Press 0 to add to the end of list\n");
    scanf("%d",&stdNo);

    p = head;
    if(p->no == stdNo){ //add to beginning
        newNode->next = p;
        head = newNode;
    }
```

Adding a Node to a List


```
    p = head;
    if(p->no == stdNo){ //add to beginning
        newNode->next = p;
        head = newNode;
    }
    else{
        while(p->next != NULL && p->no != stdNo){
            q=p;
            p= p->next;
        }
        if(p->no == stdNo){ //Add nor beginning neither end
            q->next = newNode;
            newNode->next = p;
        }
        else if(p->next == NULL){ //Add to end
            p->next = newNode;
            newNode->next = NULL;
        }
    }
    return head;
}
```

Deleting a Node from the List

- The next pointer of the item immediately preceding the one to be deleted is altered, and made to point to the item following the deleted item.



Deleting a Node from the List

- To delete a specified node (give the node whose number field is given)
- Three conditions arise:
  - Deleting the first node.
  - Deleting the last node.
  - Deleting an intermediate node.

Deleting a Node from the List


```
node* deleteNode(node *head){
    int stdNo;
    node *p, *q;

    printf("Enter student number that will be deleted?");
    scanf("%d",&stdNo);

    p = head;
    if(p->no == stdNo){ //delete node at the beginning
        head = p->next;
        free(p);
    }
```

Deleting a Node from the List


```
    p = head;
    if(p->no == stdNo){ //delete node at the beginning
        head = p->next;
        free(p);
    }
    else{
        while(p->next != NULL && p->no != stdNo){
            q=p;
            p= p->next;
        }
        if(p->no == stdNo){ //Delete from nor beginning neither end
            q->next = p->next;
            free(p);
        }
        else if(p->next == NULL){ //No node found to delete
            printf("No node found to delete");
        }
    }
    return head;
}
```

Main Function


```
int main(void)
{
    node *head;
    int selection=0;
    printf("1.Create List 2.Traverse List 3.Add Node 4.Delete Node 5.Exit\r
    while(1)
    {
        printf("Selection [1-5]?");
        scanf("%d",&selection);
        switch(selection)
        {
        case 1: head = createList();printf("Adres: %x\n",head);
            traverseList(head);break;
        case 2: traverseList(head);break;

        case 3: head=addNode(head);
            traverseList(head);break;
        case 4: head=deleteNode(head);
            traverseList(head);break;
        case 5: exit(0);

        }
    }
}
```

Singly Linked List Application-1

- A linear list application that has capability of listing nodes in alphabetical order, inserting nodes, deleting a specified node and finding the record that has maximum number of characters in the list.

Node Structure


```
 5  struct personel
 6  {
 7    char   adi[21];
 8    struct personel *sonraki;
 9  };
10
11  //artık struct personel yerine dugum kullanılacak
12  typedef struct personel dugum;
13  // *head dugum tipinde bir işaretçidir.
14  //Herzaman listenin başını gösterecek
15  dugum *head, *oncekiDugum,*yeni,*silinecek;
```

Searching Record


```
18  void ara(char *aranan) //listede arama yapar
19  {
20    dugum *p;
21    p = head;
22    oncekiDugum = head;
23    while(p->sonraki!=NULL)
24    {
25        p= p->sonraki;
26        if (strcmp(p->adi, aranan) >= 0) break;
27        oncekiDugum = p;
28    }
29  }
```

Adding a New Record


```
30  void ekle(char *s)
31  {
32    yeni = (dugum *) malloc(sizeof(dugum));
33    strcpy(yeni->adi, s);
34    ara(yeni->adi);
35    yeni->sonraki = oncekiDugum->sonraki;  /* listeye ekle */
36    oncekiDugum->sonraki = yeni;
37  }
```

Delete Record


```
38  void sil(char *s)
39  {
40    ara(s);
41    silinecek = oncekiDugum->sonraki;
42    oncekiDugum->sonraki = silinecek->sonraki;
43    free(silinecek);
44  }
```

List Nodes


```
45  void listele(void)
46  {
47      dugum *p;
48      p = head;
49      p = p->sonraki;
50      while (p!= NULL)
51      {
52        printf("%s  \n",p->adi);
53        p = p->sonraki;
54      }
55  }
```

Finding Longest Record


```
56  void enUzunBul(void)
57  {
58      dugum *p, *enuzun;
59      p = head;
60      p= p->sonraki;
61      enuzun=p;
62      while (p != NULL)
63      {
64          if (strlen(p->adi) >= strlen(enuzun->adi))
65          {
66            enuzun = p;
67          }
68          p = p->sonraki;
69      }
70      printf("\nEn uzun :%s  Uzunluk:%d",enuzun->adi,strlen(enuzun->adi));
71      getchar();
72  }
```

Main Function


```
74  void main()
75  {
76    char      sec;
77    char      s[21];
78    head = (dugum *) malloc(sizeof(dugum));
79    strcpy(head->adi," listenin basi");
80    head->sonraki = NULL;
81    do
82      {
83        system("cls");
84        listele();
85        printf("\n\n1 - Ekle\n2 - Sil\n3 - En Uzun isim\n4 - Cikis\n\nSec :");
86        sec = getche();
87        switch (sec)
88          {
89          case '1':printf("\nAdi :"); gets(s);
90            ekle(s);break;
91          case '2':printf("\nAdi ");  gets(s);
92            sil(s);break;
93          case '3':enUzunBul();break;
94          case '4':exit(0);break;
95          }
96      }
97    while (1);
98  }
```

Singly Linked List Application-2

- Do the following operations on a singly linked list application that holds several information about students such as number, name, midterm exam grade and final exam grade:
  - Add / remove record
  - Listing students with their final term grade (Midterm exam %40, Final Exam:%60)
  - List all information of the student who has highest final term grade
  - Calculate the average final term grades of the students in the list

Node Structure

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct ogrenci{
5      int no;
6      char adi[40];
7      int vize,final;
8      struct ogrenci *sonraki;
9  };
10 typedef struct ogrenci dugum;
11 dugum *head,*yeniDugum,*oncekiDugum,*enBasarili;
12
```

Creating List


```
13  void ogrenciListeOlustur()
14  {
15      int n,k;
16      printf("Kac elamanli liste olusturacaksin");
17      scanf("%d",&n);
18      for(k=0;k<n;k++)
19      {
20          if(k==0) //ilk düğüm ekleniyor
21          {
22              yeniDugum = (dugum *)malloc(sizeof(dugum));
23              head = yeniDugum;
24          }
25          else
26          {
27              yeniDugum->sonraki = (dugum *) malloc(sizeof(dugum));
28              yeniDugum = yeniDugum->sonraki;
29          }
30          printf("Ogrenci No:");scanf("%d",&yeniDugum->no);
31          printf("Ogrenci Adi:");scanf("%s",yeniDugum->adi);
32          printf("Ogrenci Vize:");scanf("%d",&yeniDugum->vize);
33          printf("Ogrenci Final:");scanf("%d",&yeniDugum->final);
34      }
35      yeniDugum->sonraki = NULL;
36  }
```

Listing Students


```
38  void ogrenciListele()
39  {
40      double donemNotu =0;
41      dugum *p;
42      p = head;
43      while(p!=NULL)
44      {
45          printf("\n ************** \n");
46          printf("Ogrenci No:%d\t",p->no);
47          printf("Ogrenci Adi:%s\t",p->adi);
48          printf("Ogrenci Vize:%d\t",p->vize);
49          printf("Ogrenci Final:%d\t",p->final);
50          donemNotu = p->vize*0.4+p->final*0.6;
51          printf("Donem Notu : %.2f\t", donemNotu);
52          printf("\n ************** \n");
53          p = p->sonraki;
54      }
55  }
```

Adding New Student


```
57  void ogrenciEkle()
58  {
59      int kayitNo;
60      dugum *p, *q;
61      yeniDugum = (dugum *) malloc(sizeof(dugum));
62      printf("Ogrenci No:");scanf("%d",&yeniDugum->no);
63      printf("Ogrenci Adi:");scanf("%s",yeniDugum->adi);
64      printf("Ogrenci Vize:");scanf("%d",&yeniDugum->vize);
65      printf("Ogrenci Final:");scanf("%d",&yeniDugum->final);
66
67      printf("Hangi kayittan oncesine eklemek istiyorsunuz");
68      printf("\nListe sonuna eklemek icin 0 gir");
69      scanf("%d",&kayitNo);
70
71      p = head;
72      if(p->no == kayitNo) //başa ekle
73      {
74          yeniDugum->sonraki = p;
75          head = yeniDugum;
76      }
```

Adding New Student


```
80      else
81      {
82          while(p->sonraki !=NULL)
83          {
84              q=p;
85              p=p->sonraki;
86              if(p->no == kayitNo) break;
87          }
88          if(p->no == kayitNo) //araya ekleme
89          {
90              q->sonraki = yeni;
91              yeni->sonraki = p;
92          }
93          else if(p->sonraki == NULL)
94          {
95              p->sonraki = yeni;
96              yeni->sonraki = NULL;
97          }
98      }
99  }
```

Delete Student


```
 97 void ogrenciSil()
 98 {
 99     int kayitNo;
100     dugum *p, *q;
101
102     printf("Silmek istediginiz ogrenci no gir");
103     scanf("%d",&kayitNo);
104
105     p = head;
106     if(p->no == kayitNo) //baştakini sil
107     {
108         head = p->sonraki;
109         free(p);
110     }
```

Delete Student


```
115     else
116     {
117         while(p->sonraki !=NULL)
118         {
119             if(p->no == kayitNo) break;
120             else {
121                 q=p;
122                 p=p->sonraki;
123             }
124         }
125         if(p->no == kayitNo) //araya ekleme
126         {
127             q->sonraki = p->sonraki;
128             free(p);
129         }
130         else if(p->sonraki == NULL)
131         {
132             printf("Silinecek ogrenci no yok\n");
133         }
134     }
135 }
```

Find Student with Highest Final Term Grade


```
130 double basariNotuHesapla(int vize,int final)
131 {
132     return (0.4*vize)+(0.6*final);
133 }
134
135 void enBasariliOgrenci()
136 {
137     double basariNotu = 0;
138
139     dugum *p;
140     p = head;
141     enBasarili = head;
142
143     while(p->sonraki!=NULL)
144     {
145         p=p->sonraki;
146         if(basariNotuHesapla(p->vize,p->final)>basariNotuHesapla(enBasarili->vize,enBasarili->final))
147             enBasarili = p;
148     }
149     printf("En basarili ogrenci:\n");
150     printf("No:%d - Ad: %s  Basari Notu:%.2f\n",enBasarili->no,
151         enBasarili->adi,    basariNotuHesapla(enBasarili->vize,enBasarili->final));
152 }
```

Calculate Final Term Average of the Class


```
154 void sinifBasariOrtalamasi()
155 {
156     double basariNotuOrtalamasi = 0;
157     double toplam =0;
158     int sayac = 0;
159     dugum *p;
160     p = head;
161     if(p==NULL)
162     {
163         printf("Listede kayit yok!");
164     }
165     else
166     {
167         do
168         {
169             toplam += basariNotuHesapla(p->vize,p->final);
170             p=p->sonraki;
171             sayac++;
172         }
173         while(p!=NULL);
174 
175         basariNotuOrtalamasi = toplam/sayac;
176         printf("Basari Notu Ortalamasi: %.2f \n",basariNotuOrtalamasi);
177     }
178 }
```

Main Function


```
180 int main(void)
181 {
182     int secim=0;
183     printf("1-Liste Olustur \n2-Yeni Kayit Ekle \n3-Kayit Sil
184     while(1)
185                                         \n4-En Yuksek Basari Notu\n5-Sinif Ortalamasi\n");
186     {
187         printf("Secim yap [1-5]?");
188         scanf("%d",&secim);
189         switch(secim)
190         {
191         case 1: ogrenciListeOlustur();
192             ogrenciListele();break;
193         case 2: ogrenciEkle();ogrenciListele();break;
194         case 3: ogrenciSil();ogrenciListele();break;
195         case 4: enBasariliOgrenci();break;
196         case 5: sinifBasariOrtalamasi();break;
197         }
198     }
}
```

Next Week

- File Operations
- Sequential Access Files

References

- Doç. Dr. Fahri Vatansever, “Algoritma Geliştirme ve Programlamaya Giriş”, Seçkin Yayıncılık, 12. Baskı, 2015.
- Kaan Aslan, “A’dan Z’ye C Klavuzu 8. Basım”, Pusula Yayıncılık, 2002.
- Paul J. Deitel, “C How to Program”, Harvey Deitel.
- “A book on C”, All Kelley, İra Pohl

Any Questions ?

Thanks for listening

CANER ÖZCAN
canerozcan@karabuk.edu.tr

Footnote (Turkish)..

"Hayat C programı gibidir. Main de başlarsın herşeye, while da takılır kalırsın bazen, sonra sinirlenip break dersin. Kimi zaman fonksiyonlar oyalar seni ama sonunda return'ü bulur kaçarsın. Bazen if dersin olmaz, else if dersin o da olmaz, aradığını else te bulursun. Switch dersin, case dersin, sonunda defaultu yersin... Derdini anlatmaya char, int yetmez, long dersin. Ama teferruatını float dinler. O da globaldir gider herkese anlatır. Laftan anlamayana decimal konuşursun. Seni anlayana hexadecimal... Gün gelir dizinin son karakteri olursun ama döngüyü sen bitirisin. Sprintf dersin haykırırsın dünyaya, isyanın stringe sığmaz, derleyene kalır..."