Matemàtica computacional i analítica de dades
Algorítmia i combinatòria en grafs...
Curs 2020–21

5 Treball amb llistes enllaçades

El codi d'exemple següent llegeix la mateixa *llista de notes* de les Pràctiques 2 i 4 en forma de llista enllaçada en la que les dades de cada línia són un identificador numèric (NIU) i quatre valors decimals entre 0 i 10, mentre es va fent la lectura es calcula la mitjana d'aquests quatre valors i es guarda tota aquesta informació en un node d'aquesta llista. Un detall addicional és que, cada cop que s'incorpora informació nova a la llista, es busca el lloc on fer aquesta incorporació per tal que les dades quedin ordenades seguint l'ordre de l'identificador. D'aquesta manera, treballem des del principi amb una llista ordenada, i es fa una manipulació més fàcil quan aquesta llista correspon a una cua o a una pila.

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

typedef struct Dada{
    int niu;
    float notes[5];
    struct Dada * seg;
} Alu;

float mitjana(float [],int );

int main(){
    FILE *dades;
    int n,i,lrg=0;

    Alu *inicill=NULL, *actual = NULL, *anterior;

    dades=fopen("Llista.txt","r");
    if(dades==NULL){
        printf("\nNo s'ha accedit al fitxer de dades\n");
        return 1;
    }

    while(!(fscanf(dades,"%i;",&n)==EOF)){
        if((actual=(Alu *)malloc(sizeof(Alu)))==NULL)
        {
            printf("Problema assignant espai de memoria\n");
            return 2;
        }
        lrg++;
        actual->niu=n;
        for(i=0;i<4;i++){
            fscanf(dades,"%f",&actual->notes[i]);
            fgetc(dades);//llegeix i descarta els ; i el \n
        }
        actual->notes[4]=mitjana(actual->notes,4);
        if(inicill==NULL){
            actual->seg=NULL;
            inicill=actual;
        }
        else{
            if (actual->niu<inicill->niu){
                actual->seg=inicill;
                inicill=actual;
```

```
45              }
46              else{
47                  anterior=inicill;
48                  while(anterior->seg!=NULL&&(anterior->seg)->niu<actual->niu
     ){
49                      anterior=anterior->seg;
50                  }
51                  actual->seg=anterior->seg;
52                  anterior->seg=actual;
53              }
54          }
55
56      }
57      fclose(dades);
58      actual=inicill;
59      while(!(actual==NULL)){
60          printf("%d | ",actual->niu);
61          for(i=0;i<4;i++){
62              printf("%5.1f",actual->notes[i]);
63          }
64          printf(" |%6.2f",actual->notes[4]);
65          printf("\n");
66          actual=actual->seg;
67      }
68      printf("\nS'ha llegit informacio de %d linies.\n\n",lrg);
69
70      return 0;
71  }
72
73  float mitjana(float dades[],int n){
74      int i;
75      float m=0.;
76      for(i=0;i<n;i++)
77      {
78          m+=dades[i];
79      }
80      return m/n;
81  }
```

5.1 Exercicis

Afegiu com a les dues primeres línies del programa i en format comentari els vostres `Nom`, `Cognom` i `NIU`.

El nom dels programes que contenen el codi ha de ser de la forma `PrXExY.c`, on `X` fa referència a la pràctica i `Y` a l'exercici. Per exemple, el nom del programa de l'apartat següent hauria de ser `Pr5Ex511.c`.

Exercici 5.1.1: Feu una funció `imprimirllista` que, a partir dels arguments que siguin necessaris, imprimeixi la llista en el mateix format que fa aquest programa. Modifiqueu el `main` inicial substituint la part que mostra la sortida per pantalla per la funció `imprimirllista`.

Exercici 5.1.2: Convertiu el bloc que insereix la informació que s'ha llegit d'una línia en una funció `inserir` (amb els arguments adequats) i utilitzeu-la per incloure al programa l'opció d'afegir manualment un element nou a la llista (el programa ha de preguntar per pantalla el valor del NIU i les 4 notes i, mitjançant la funció que heu programat, inserir el cas a la posició adequada).
Per comprovar-ne el funcionament, inseriu l'alumne `1234567` amb notes `5.6 6.2 4.3 8.6` i mostreu per pantalla la nova llista.

Exercici 5.1.3: En el sentit invers de l'anterior, contempleu l'opció d'eliminar la informació corresponent a un identificador donat mitjançant una funció `esborrar`. Caldrà localitzar el bloc corresponent a l'identificador, refer la llista i alliberar la memòria. Per comprovar-ne el funcionament, esborreu (si existeixen) les notes amb NIU `1000961`, `1110847`, `3900285` i `3989795`. Imprimiu per pantalla la llista després d'aquestes modificacions.

5.2 `qsort` amb llistes enllaçades

A aquest apartat hauríeu de poder aprofitar les funcions que heu programat a la Pràctica 4.

Si volem mostrar els elements d'una llista enllaçada en un ordre diferent, no cal que reordenem la llista enllaçada, si no que podem crear vectors d'índex on cada element del vector apunti a la posició de memòria d'un element de la llista enllaçada. Si seguim l'ordre que ens dóna aquest vector, podrem accedir als elements de la llista en aquest ordre i, en particular, obtenir un llistat ordenat en un ordre diferent al que estan guardades les dades.

En aquest cas, la funció `qsort` rebrà aquest vector i el reordenarà. Per altra banda, la funció `comparacio` haurà de llegir cada component d'aquest vector, que serà l'adreça d'un alumne. Llavors, a partir de dues posicions diferents d'aquest vector, haurà de retornar `1` si la primera és més gran que la segona i `-1` altrament.

Exercici 5.2.1: Definiu la variable `adrecesordremitjana` com un punter de punters tipus `Alu`, reserveu la memòria necessària i guardeu a cada posició l'adreça de cada alumne en l'ordre que ens dóna la llista enllaçada.

Feu també una funció `imprimirenordre` que, a partir del vector de les adreces (i el nombre d'alumnes) mostri el llistat d'abans en aquest ordre.

Feu una funció `comparaciomitjana` que retorni `1` o `-1` depenent de si la nota mitjana del primer alumne és més gran que la del segon o no.

Ordeneu el vector `adrecesordremitjana` amb el criteri `comparaciomitjana` i mostreu el nou ordre per pantalla.

Observació: Heu de tenir en compte que, si voleu que el programa continuï sent consistent i hi ha les funcions que insereixen i esborren elements a la llista, o bé aquestes han d'afegir i treure l'índex corresponent, o bé s'ha de refer el vector `adrecesordremitjana` després de cada modificació de la llista.

Exercici 5.2.2: El nombre d'índexs que es poden associar a una llista és arbitrari. Fins i tot si la llista ja està ordenada per l'identificador pot ser interessant mantenir un índex basat en aquest camp de cada bloc: pot ser útil per anar a una posició particular sense haver de recórrer tota la llista. En particular, si teniu un vector `indexniu` tal que `indexniu[i]` tingui la posició de memòria que ocupa l'`i`-èssim element de la llista (que ja està ordenada per NIU), podreu realitzar les cerques d'un identificador dins la llista de forma binària i no seqüencial.

Modifiqueu el programa per a que pregunti un NIU i mostri la informació de les notes corresponents a aquest NIU fent una recerca binària utilitzant `indexniu` (que haureu d'haver definit abans).

Instruccions finals

Quan acabeu la pràctica, feu el lliurament dels fitxers de codi (que tenen els noms de la forma `Pr5ExY.c` segons el que hem indicat anteriorment) a través del Campus Virtual des de l'apartat de lliuraments de l'assignatura.