Serialização de Estruturas de dados em C

Recordando o Módulo de Listas

```
#ifndef _LL_GENERIC
#define _LL_GENERIC
int init (int size, int (*compare)(void *, void *));
int insert (int handle, void * data);
int search (int handle, void *data);
int remov (int handle, void *data);
int clean (int handle);
#endif
```

Persistência

```
#ifndef _LL_GENERIC
#define _LL_GENERIC
int init (int size, int (*compare)(void *, void *));
int insert (int handle, void * data);
int search (int handle, void *data);
int remov (int handle, void *data);
int clean (int handle);
int write_to(int handle, char* filename);
int read_from(int handle, char* filename);
#endif
```

```
int write_to(int handle, char* filename);
```

- Abrir ficheiro
- Fazer travessia da lista
  - para cada elemento escrever para o ficheiro

int write_to(int handle, char* filename);

```
int write_to(int handle, char* filename){
    FILE* f;
    Node* act;

    if (!valid_handle(handle)) return 0;

    f=fopen(filename,"w+");
    if(f==NULL){
        perror("fopen");
    }

    act = control[handle].root;

    while (act!=NULL) {
        int cnt;
        cnt=fwrite(act->data, control[handle].sizeofData , 1,f);
        if(cnt <1) perror(“Write to file”);
        act = (Node *)act->next;
    }
    fclose(f);
    return 0;
}
```

E se o bloco de dados tiver apontadores?

- Função específica para escrever o tipo de dados para o ficheiro

```
int write_to(int handle, char* filename,
    void (*mydata_write_to)(void *, FILE* ));
```

- Função específica para serializar o tipo de dados

```
int write_to(int handle, char* filename,
    void* (*mydata_write_to)(void *, int* ));
```

int read_from(int handle, char* filename);

- Abrir ficheiro
- Ler do ficheiro os elementos e inserir na lista

int read_from(int handle, char* filename);
V1

```
int read_from2(int handle, char* filename){
    FILE* f;  Node* act;  int size;    void* buffer;
    if (!valid_handle(handle)) return 0;
    f=fopen(filename,"r");
    if(f==NULL){
        perror("open");
        exit(-1);
    }
    act = control[handle].root;
    int data_size=control[handle].sizeofData;
    buffer=malloc(data_size);
    while(fread(buffer,data_size,1,f)){
        insert_aux(handle,buffer,1);
        buffer=malloc(data_size);
    }
    free(buffer);
return 0;
}
```

int read_from(int handle, char* filename);
V2

```
int read_from3(int handle, char* filename){
    FILE* f;  Node* act;
    int size;   void* buffer;
    if (!valid_handle(handle)) return 0;
    // Abrir o ficheiro
    f=fopen(filename,"r");
    if(f==NULL){
        perror("open");
        exit(-1);
    }
    // calcular o tamanho do ficheiro
    struct stat st;
    stat(filename, &st);
    size = st.st_size;
    act = control[handle].root;
    // Ler o ficheiro para memória
    buffer=malloc(size);
    fread(buffer,size,1,f);
    fclose(f);
    int data_size=control[handle].sizeofData;
    // percorrer o buffer como um array
    // e inserir os elementos na lista
    void *aux=buffer;
    while (aux != buffer+size){
        insert_aux(handle,aux,1);
        aux+=data_size;
    }
    return 0;
}
```

int read_from(int handle, char* filename);
V3 (otimização por os elementos estarem ordenados)

```
int read_from3(int handle, char* filename){
    FILE* f;  Node* act;
    int size;   void* buffer;
    if (!valid_handle(handle)) return 0;
    // Abrir o ficheiro
    f=fopen(filename,"r");
    if(f==NULL){
        perror("open");
        exit(-1);
    }
    // calcular o tamanho do ficheiro
    struct stat st;
    stat(filename, &st);
    size = st.st_size;
    act = control[handle].root;
    // Ler o ficheiro para memória
    buffer=malloc(size);
    fread(buffer,size,1,f);
    fclose(f);
    int data_size=control[handle].sizeofData;
    // percorrer o buffer como um array
    // e inserir os elementos na lista
    void *aux=buffer+(size-data_size);
    while (aux > buffer){
        insert_aux(handle,aux,1);
        aux-=data_size;
    }
    return 0;
}
```

Exemplo

- IN:Ficheiro de utilizadores em texto (formato DOS)

9000011000	Jon Yang	9000011000@foursquareUM.com	3761 N. 14th St
9000011001	Eugene Huang	9000011001@foursquareUM.com	2243 W St.
9000011002	Ruben Torres	9000011002@foursquareUM.com	5844 Linden Land
9000011003	Christy Zhu	9000011003@foursquareUM.com	1825 Village Pl.

- OUT: Ficheiro de utilizadores em binário

Exemplo

```
typedef struct {
    char id[15];
    char name[80];
    char email[80];
    char morada[80];
} My_Data;

int compara (void *a, void *b) {
        My_Data *aa, *bb;
        aa = (My_Data*) a;
        bb = (My_Data*) b;
        return strcmp(aa->id , bb->id);
}
```

```
int main (int argc, char **argv) {
    int op, LL;
    My_Data d;
    FILE * f;
    char *buffer,*b;
    buffer=malloc(200*sizeof(char));
    b=buffer;
    if(argc <2){
        printf("Missing argument <filename>");
        exit(-1);
    }

f=fopen(argv[1],"r");
if(f==NULL){
    perror("open");
    exit(-1);
}
```

Exemplo

```
int main (int argc, char **argv) {
    int op, LL;
    My_Data d;
    FILE * f;
    char *buffer,*b;
    buffer=malloc(200*sizeof(char));
    b=buffer;
    if(argc <2){
        printf("Missing argument
                <filename>");
        exit(-1);
    }

    f=fopen(argv[1],"r");
    if(f==NULL){
        perror("open");
        exit(-1);
    }
```

```
    while(fgets(buffer,200,f)!=NULL){
        int t=strlen(buffer);
        buffer[t-2]='\0';
        buffer[t-1]='\0';
        My_Data *usr=(My_Data*)malloc(sizeof(My_Data));
        char *token;
        token=strsep(&buffer,"\t");
        memcpy(usr->id,token,strlen(token)+1);
        token=strsep(&buffer,"\t");
        memcpy(usr->name,token,strlen(token)+1);
        token=strsep(&buffer,"\t");
        memcpy(usr->email,token,strlen(token)+1);
        token=strsep(&buffer,"\t");
        memcpy(usr->morada,token,strlen(token)+1);
        insert(LL,usr);
        buffer=b;
        memset(b,0,200*sizeof(char));
    }
    write_to(LL,"usr.db");
    return ;
}
```