

Ayudantía Tarea1 ILI-135

Recuperación de Información

1º Semestre 2008

Boris Bustos Castillo

blbustos@alumnos.inf.utfsm.cl

Valparaíso 18 de Marzo, 2008

1. Acceso a los Archivos.

Se refiere al método utilizado para acceder a los registros de un archivo prescindiendo de su organización. Existen distintas formas de acceder a los datos:

- Archivos binarios: archivos guardados en una codificación básica para el computador, esta puede ser de 0's y 1's, pero no es texto plano.
- Archivos secuenciales: los registros se leen desde el principio hasta el final del archivo, de tal forma que para leer un registro se leen todos los que preceden.
- Archivos Directos: cada registro puede leerse/escribirse de forma directa solo con expresar su dirección en el fichero por el número relativo del registro o por transformaciones de la clave de registro en el número relativo del registro a acceder.

2. Funciones Útiles en el Desarrollo de la Tarea.

- Abrir un Archivo:

```
FILE* fopen(char *nombre, char *modo);  
*nombre = nombre del archivo o ruta  
*modo = "rb", "wb", "ab", "rb+"
```

- Cerrar un Archivo:

```
int fclose(FILE* ptr); // retorna 0 si se cierra con éxito
```

- Acceder a un registro de forma directa:

```
int fseek(FILE *ptr, long int desplazamiento, int origen); // retorna 0 si se ejecuta con éxito
```

desplazamiento = cantidad de caracteres en bytes desde una posición indicada por el flag origen.
origen = SEEK_SET, SEEK_CUR, SEEK_END.

- Lectura de un registro:

```
size_t fread(void *estructura, size_t tamanyo, size_t n_estruct, FILE *ptr);  
// retorna una estructura leída a *estructura y 0 si es EOF.
```

```
tamanyo = sizeof(estructura);
n_estruct = numero de estruct a leer.
```

- Escritura de un registro:

```
size_t fwrite(void *estructura, size_t tamanyo, size_t n_estruct, FILE *ptr);
```

- Resetear el puntero al comienzo de un archivo:

```
rewind(FILE *ptr);
```

3. Ayuda en la Implementación de los Archivos.

- Personal.ari = archivo binario relativo, de acceso directo.

```
typedef struct{
    char rut [10];
    char nombre[100];
    char direccion[100];
    char telefono[10];
    int edad
}Personal;

FILE *fichero;
Personal nuevo_pers;

if((fichero=fopen("personal.ari", "rb+"))==NULL){
    printf("error al abrir el archivo personal.ari\n");
}

fseek(fichero,(nuevo_pers.rut%100)*sizeof(Personal),SEEK_SET);

fread(&nuevo_pers,sizeof(Personal),1,fichero);
fwrite(&nuevo_pers,sizeof(Personal),1,fichero);

fclose(fichero);
```

- habitacion.ari = archivo binario de acceso secuencial y largo variable.

```
typedef struct{
    int num_hab; // 11, 12, 21, 22, etc.
    char tipo[12]; // single = 1 cama, doble = 2 camas, triple = 3 camas, matrimonial = 1 cama.
    int pension; // 1 = con servicio de alimentación las 24 horas, 0 = sin ese servicio exclusivo.
    char observacion[50];
}Habitacion;

typedef struct{
    int id;
    float dimension; // 1 = 1 plaza, 1.5 = plaza y media, 2 = 2 plazas.
    int valor_diario; // 5000 =cama 1 plaza, 7000 = cama plaza y media, 15000 = cama 2 plazas.
}Cama;

FILE *fichero;
Int contador=0, i;
Habitacion nueva_hab;
Cama nueva_cama;
```

```
if((fichero=fopen("habitacion.ari", "rb+"))==null){
    printf("error");
}

while(fread(&nueva_hab , sizeof(Habitacion),1,fichero)){

    // código

    if ( strcmp(nueva_hab.tipo, "single") == 0) contador =1;
    if ( strcmp(nueva_hab.tipo, "doble") == 0) contador =2;
    if ( strcmp(nueva_hab.tipo, "triple") == 0) contador =3;
    if ( strcmp(nueva_hab.tipo, "matrimonial") == 0) contador =1;

    for(i=0;i<contador;i++){
        fread(&nueva_cama , sizeof(Cama),1,fichero);

        // código
    }
}
```