Cartilla de Organizaciones de Archivos

1.- Streams.

Se utilizarán las funciones provistas por el lenguaje.

2.- Organización Secuencial.

```
int S_CREATE (const char* nombre_fisico);
int S_OPEN (const char* nfisico, int modo);
int S_CLOSE (int handler);
int S_READ (int handler, void* reg);
int S_WRITE (int handler, const void* reg, unsigned long cant);
int S_DESTROY(const char* nombre_fisico);
```

3.- Organización Relativa.

```
int R_CREATE (const char* nombre_fisico, int tam_registro, int max_reg);
int R_OPEN (const char* nombre_fisico, int modo);
int R_CLOSE (int handler);
int R_SEEK (int handler, int nrec);
int R_READ (int handler, int nrec, void* reg);
int R_READNEXT (int handler, void* reg);
int R_WRITE (int handler, int nrec, const void* reg);
int R_UPDATE (int handler, int nrec, const void* reg);
int R_DELETE (int handler, int nrec);
int R_DESTROY(const char* nombre_fisico);
int R_GETMAXREGS( int handler );
```

4.- Organización Directa.

```
int D_CREATE (const char* nombre_fisico, const campo *reg, const campo *clave,  int
max_reg);
int D_OPEN (const char* nombre_fisico, int modo);
int D_CLOSE (int handler);
int D_READ (int handler,  void* reg);
int D_WRITE (int handler, const void* reg);
int D_UPDATE (int handler, const void* reg);
int D_DELETE (int handler, const void* reg);
int D_DESTROY(const char* nombre_fisico);
```

5.- Organización Indexada.

```
int I_CREATE (const char* nfisico, const campo *reg, const campo *cl_prim);
int I_OPEN (const char* nombre_fisico, int modo);
int I_CLOSE (int handler);
int I_ADD_INDEX(int handler, const campo *cl);
int I_DROP_INDEX(int handler, int indexId);
int I_IS_INDEX (int handler, campo *clave);
int I_START(int handler, int indexId, char* operador, const void* val_ref);
int I_READ (int handler,  void* reg);
int I_READNEXT(int handler, int indexId, void* reg);
int I_WRITE (int handler, const void* reg);
```

```
int I_UPDATE (int handler, const void* reg);
int I_DELETE (int handler, const void* reg);
int I_DESTROY(char* nombre_fisico);
```

6.-Definiciones de las primitivas

```
typedef struct {
      const char* nombre;
      int   tipo;
      int   longitud;
}campo;

/* valores con signo. */
#define CHAR      1
#define INT       2
#define LONG      3
#define FLOAT     4
#define DOUBLE    5
#define UNSIGNED  0x80


#define READ                1
#define WRITE               2
#define READ_WRITE          3 /* READ | WRITE */
#define APPEND              4
```

7.- Valores devueltos por las primitivas

```
#define OK                   0
#define NULL                -1
#define ERROR               -2
#define EOF                 -3
#define EXISTE              -4
#define NO_EXISTE           -5
#define EXISTE_INDICE       -6
#define NO_ EXISTE_INDICE   -7
#define ARCHIVO_LLENO       -8
#define ES_PRIM             -9
```