```
#ifndef ARCHIVOTEMPORAL
#define ARCHIVOTEMPORAL
#include <iostream.h>
#include <stdio.h>
#include <stdlib.h>
#define ERR -1
#define OK 0

class ArchivoTemporal
{
   int tam_reg;
   char tempfich[15];
   int num_reg;
   void CrearATMP(void);
   int CopiarATMP(const ArchivoTemporal &);
   int Rellenar(int);
   public:
   ArchivoTemporal(int);
   ~ArchivoTemporal() { unlink(tempfich); }
   ArchivoTemporal(const ArchivoTemporal &);
   int Escribir(int, void *);
   int Leer (int, void *) const;

};

int ArchivoTemporal::CopiarATMP(const ArchivoTemporal &AT)
{
   FILE *f1, *f2;
   char *buffer;

   if(tam_reg!=AT.tam_reg) return ERR;
   buffer=new char [tam_reg];
   f1=fopen(tempfich, "wb");
   if(!f1) return ERR;
   f2=fopen(AT.tempfich, "rb");
   if(!f2) return ERR;
   fread(buffer, tam_reg, 1, f2);
   while (!feof(f2))
   {
      fwrite(f1, tam_reg, 1, f1);
      fread(buffer, tam_reg, 1, f2);
   }
   delete []buffer;
   fclose(f1);
   fclose(f2);
   return OK;
}

ArchivoTemporal::ArchivoTemporal(int tam)
{
   tam_reg=tam;
   num_reg=0;
   CrearATMP();
}

void ArchivoTemporal::CrearATMP(void)
{
   int correcto=0;
   FILE *fp;

   randomize();
   do
   {
      sprintf(tempfich, "ATMP%d.tmp", rand()%10000);
      fp=fopen(tempfich, "r");
      if(!fp)
      {
         fp=fopen(tempfich, "w");
         correcto=1;
      }
      fclose(fp);
   }
   while(!correcto);
}
```

```
ArchivoTemporal::ArchivoTemporal(const ArchivoTemporal &AT)
{
   tam_reg=AT.tam_reg;
   num_reg=AT.num_reg;
   CrearATMP();
   CopiarATMP(AT);
}

int ArchivoTemporal::Escribir(int pos, void *reg)
{
   FILE *fp;
   if (pos>num_reg)
      Rellenar(pos);
   fp=fopen(tempfich, "r+b");
   if(!fp) return ERR;
   fseek(fp, pos*(long)tam_reg, SEEK_SET);
   fwrite(reg, tam_reg, 1, fp);
   fclose(fp);
   return OK;
}

int ArchivoTemporal::Rellenar (int hasta)
{
   FILE *fp;
   char *buffer;

   buffer=new char [tam_reg];
   memset(buffer, ' ', tam_reg);
   fp=fopen(tempfich, "ab");
   for (int i=num_reg; i<=hasta; i++)
      fwrite(buffer, tam_reg, 1, fp);
   fclose(fp);
   delete []buffer;
   num_reg=hasta+1;
   return OK;
}

int ArchivoTemporal::Leer(int pos, void *reg)  const
{
   FILE *fp;

   if (pos>num_reg) return ERR;
   fp=fopen(tempfich, "rb");
   fseek(fp, pos*(long)tam_reg, SEEK_SET);
   fread(reg, tam_reg, 1, fp);
   fclose(fp);
   return OK;
}

// #define ARCHIVOTEMPORAL_TEST
#ifdef ARCHIVOTEMPORAL_TEST
#define MAX_FRASE 30
void Escritura (ArchivoTemporal& A);
void Lectura (ArchivoTemporal &A);

void main(void)
{
   ArchivoTemporal A(MAX_FRASE);
   int seguir=1;


   while(seguir)
   {
      cout<<"0. Salir"<<endl;
      cout<<"1. Escribir"<<endl;
      cout<<"2. Leer"<<endl;
      cin>>seguir;
      switch(seguir)
      {
         case 1: Escritura(A); break;
         case 2: Lectura(A);   break;
      }
   }
```

```
}

void Escritura (ArchivoTemporal& A)
{
   char frase [MAX_FRASE];
   int posicion;

   cout<<"Deme una frase: ";
   cin.ignore();
   cin.get(frase,MAX_FRASE);
   cout<<"¿En qué posición la guardamos?: ";
   cin>>posicion;
   A.Escribir(posicion, frase);
}

void Lectura (ArchivoTemporal &A)
{
   char frase [MAX_FRASE];
   int posicion;

   cout<<"Posicion de lectura: ";
   cin>>posicion;
   A.Leer(posicion, frase);
   cout<<"La frase leida es:"<<frase<<endl;
}
#endif
#endif
```