Carrera: Técnico Universitario en Programación
Materia: Programación II
Tema: Datos y consignas para resolver el final

Final Programación II

Duración 60 minutos

Contenido general
El parcial consta de 3 preguntas que deben resolverse escribiendo el código correspondiente en un único archivo fuente en CodeBlock (cpp) de nombre nombreAlumno.cpp. Los puntos a responder se obtienen en el cuestionario **Cuestionario Final**.

Una vez finalizado el trabajo el archivo con la solución de los puntos debe subirse por medio del link **Entrega final**. Para ser corregido el archivo **DEBE COMPILAR.**

El tiempo efectivo de trabajo, una vez aclaradas las dudas sobre el funcionamiento, es de **1 hora**. **Pasado ese tiempo no puede entregarse el examen, por lo que se considerará como desaprobado.**

Todas los puntos en los que se solicite que se resuelva algo, y que puedan ser ejecutados, deben ser ejecutados por el programa que se entregue. Pueden ser desde un menú, o desde las funciones llamadas por main().

En el caso de que existan preguntas teóricas, desarrollarlas y dejarlas comentadas en el cpp

Para el desarrollo de la solución de los puntos se debe:

- **Utilizar la clase ArchivoFinal, que se encuentra más abajo en este documento (copiarla en el cpp).**

- **Desarrollar la clase necesaria para el manejo de los registros del archivo que se pide en los puntos a resolver. Ponerle Final como nombre a esta clase.**

- **Para la clase Final sólo hay que hacer los métodos necesarios para el desarrollo de los puntos.**

- **Para probar el programa puede usarse el archivo final.dat que se puede bajar desde Entrega Final**

Clase ArchivoFinal

```
class ArchivoFinal{
   private:
      std::string _nombreArchivo;

   public:
      ArchivoFinal(std::string nombreArchivo);
      bool Guardar(Final final);
      bool Guardar(Final final, int posicion);
      int Buscar(int ID);
      Final Leer(int posicion);
      int CantidadRegistros();
};



ArchivoFinal::ArchivoFinal(std::string nombreArchivo){
   _nombreArchivo = nombreArchivo;
}

bool ArchivoFinal::Guardar(Final final){
   FILE *pArchivo = fopen(_nombreArchivo.c_str(), "ab");
   if(pArchivo == NULL){
      return false;
   }
   bool ok = fwrite(&final, sizeof(Final), 1, pArchivo);
   fclose(pArchivo);
   return ok;
}

bool ArchivoFinal::Guardar(Final final,int posicion){
   FILE *pArchivo = fopen(_nombreArchivo.c_str(), "rb+");
   if(pArchivo == NULL){
      return false;
   }
   fseek(pArchivo, sizeof(Final) * posicion, SEEK_SET);
   bool ok = fwrite(&final, sizeof(Final), 1, pArchivo);
   fclose(pArchivo);
   return ok;
}

int ArchivoFinal::Buscar(int ID){
   FILE *pArchivo = fopen(_nombreArchivo.c_str(), "rb");
   if(pArchivo == NULL){
      return -1;
   }
  Final final;
```

```
    int i = 0;
    while(fread(&final, sizeof(final), 1, pArchivo)){
        if(final.getID() == ID){
            fclose(pArchivo);
            return i;
        }
        i++;
    }
    fclose(pArchivo);
    return -1;
}

Final ArchivoFinal::Leer(int posicion){
    Final final;
    FILE *pArchivo = fopen(_nombreArchivo.c_str(), "rb");
    if(pArchivo == NULL){
        return final;
    }

    fseek(pArchivo, sizeof(Final) * posicion, SEEK_SET);
    fread(&final, sizeof(Final), 1, pArchivo);
    fclose(pArchivo);
    return final;
}

int ArchivoFinal::CantidadRegistros(){
    FILE *pArchivo = fopen(_nombreArchivo.c_str(), "rb");
    if(pArchivo == NULL){
        return 0;
    }
    fseek(pArchivo, 0, SEEK_END);
    int cantidadRegistros = ftell(pArchivo) / sizeof(Final);
    fclose(pArchivo);
    return cantidadRegistros;
}
```