# *Unidad Nº6: Flujos y excepciones*

## *Preguntas orientadoras*

1) ¿Uno de los principales objetivos de los flujos (streams) es: _____?.

2) Escribir en el disco (y en la pantalla, aunque en menor extensión) es muy "costoso". Lleva mucho tiempo (relativamente hablando) escribir información en el disco o leer información del disco, y la ejecución del programa por lo general se bloquea debido a las lecturas y escrituras de disco. ¿Cómo se soluciona este problema?

3) Como es de esperarse, C++ se basa en el método orientado a objetos para implementar los flujos y los búffers. Explique el objetivo de las siguientes clases:

- streambuf
- ios
- istream y ostream
- iostream
- fstream

4) ¿Qué es el operador de extracción, y qué hace?

5) ¿Cuáles son las tres formas de utilizar *cin.get()*, y cuáles son sus diferencias?

6) ¿Cuál es la diferencia entre *cin.read()* y *cin.getline()*?

7) ¿Cuál es el ancho predeterminado para enviar como salida un entero largo mediante el operador de inserción?

8) ¿Cuál es el valor de retorno del operador de inserción?

9) ¿Qué parámetro lleva el constructor para un objeto *ofstream*?

10) ¿Qué hace el argumento *ios::ate*?

11) Cuando inicia un programa de C++ que incluye la clase iostream, se crean e inicializan cuatro objetos de E/S estándar. ¿Cuáles son?

12) ¿Qué es una excepción?

13) ¿Qué es un bloque *try*?

14) ¿Qué es una instrucción *catch*?

15) ¿Qué información puede contener una excepción?

16) ¿Cuándo se crean los objetos de excepción?

17) ¿Se deben pasar las excepciones por valor o por referencia?

18) ¿Atrapará una instrucción *catch* una excepción derivada si está buscando la clase base?

19) ¿Qué significa *catch(...)*?

20) ¿Por qué preocuparse por producir excepciones? ¿Por qué no manejar el error donde ocurre?

21) ¿Por qué generar un objeto?¿Por qué no sólo pasar un código de error?

22) ¿Se tiene que atrapar una excepción en el mismo lugar en el que el bloque *try* la creó?

## *Ejercicios*

1) Explique cada línea del siguiente fragmento de código:

```
double leerDouble()
{
    cin.exceptions(ios::failbit | ios::badbit);

    double dato = 0.0;
    try
    {
        cin >> dato;
    }
    catch(ios_base::failure & e)
    {
        cout << e.what() << ": dato no válido" << endl;
        cin.clear();
        cin.ignore(numeric_limits<int>::max(), '\n');
    }
    return dato;
}
```

2) Cree un bloque *try*, una instrucción *catch* y una excepción simple.
3) Modifique la respuesta del ejercicio 2, coloque datos en la excepción junto con una función getter, y utilícela en el bloque *catch*.
4) Modifique la clase del ejercicio 3 para que sea una jerarquía de excepciones. Cambie el bloque *catch* para utilizar los objetos derivados y los objetos base.
5) Escriba un programa que tome un nombre de archivo como parámetro y que abra el archivo para lectura. Lea todos los caracteres del archivo y despliegue en la pantalla sólo las letras y los signos de puntuación. (Ignore todos los caracteres no imprimibles).
6) Escriba un programa que tome un nombre de archivo como parámetro y que abra el archivo para lectura. Lea todos los bytes y presente en pantalla la información en formato Hex, como se muestra a continuación. Además, genere un nuevo archivo de texto y guarde la información presentada en

pantalla, respetando el mismo formato.

```
00000000: 25 50 44 46 2D 31 2E 34|0D 25 E2 E3 CF D3 0D 0A | %PDF-1.4%âãÏÓ
00000010: 32 37 32 34 20 30 20 6F|62 6A 0D 3C 3C 2F 4C 69 | 2724 0 obj<</Li
00000020: 6E 65 61 72 69 7A 65 64|20 31 2F 4C 20 34 33 31 | nearized 1/L 431
00000030: 31 32 34 2F 4F 20 32 37|32 37 2F 45 20 38 39 32 | 124/O 2727/E 892
00000040: 35 33 2F 4E 20 34 38 2F|54 20 33 37 36 35 39 35 | 53/N 48/T 376595
00000050: 2F 48 20 5B 20 36 33 32|20 35 39 35 5D 3E 3E 0D | /H [ 632 595]>>
00000060: 65 6E 64 6F 62 6A 0D 20|20 20 20 20 20 20 20 20 | endobj
00000070: 20 20 0D 0A 78 72 65 66|0D 0A 32 37 32 34 20 31 |   xref2724 1
00000080: 36 0D 0A 30 30 30 30 30|30 30 30 31 36 20 30 30 | 60000000016 00
00000090: 30 30 30 20 6E 0D 0A 30|30 30 30 30 30 31 34 35 | 000 n000000145
000000A0: 32 20 30 30 30 30 30 20|6E 0D 0A 30 30 30 30 30 | 2 00000 n00000
000000B0: 30 31 37 39 39 20 30 30|30 30 30 20 6E 0D 0A 30 | 01799 00000 n0
000000C0: 30 30 30 30 30 31 39 34|36 20 30 30 30 30 30 20 | 000001946 00000
000000D0: 6E 0D 0A 30 30 30 30 30|30 32 32 38 37 20 30 30 | n0000002287 00
000000E0: 30 30 30 20 6E 0D 0A 30|30 30 30 30 30 32 38 38 | 000 n000000288
000000F0: 36 20 30 30 30 30 30 20|6E 0D 0A 30 30 30 30 30 | 6 00000 n00000
00000100: 30 32 39 32 35 20 30 30|30 30 30 20 6E 0D 0A 30 | 02925 00000 n0
00000110: 30 30 30 30 30 33 30 30|34 20 30 30 30 30 30 20 | 000003004 00000
00000120: 6E 0D 0A 30 30 30 30 30|30 33 37 37 32 20 30 30 | n0000003772 00
00000130: 30 30 30 20 6E 0D 0A 30|30 30 30 30 30 34 30 30 | 000 n000000400
00000140: 35 20 30 30 30 30 30 20|6E 0D 0A 30 30 30 30 30 | 5 00000 n00000
00000150: 30 34 37 33 32 20 30 30|30 30 30 20 6E 0D 0A 30 | 04732 00000 n0
00000160: 30 30 30 30 30 34 39 37|32 20 30 30 30 30 30 20 | 000004972 00000
```

7) Realizar un programa que permita crear un archivo nuevo, abrir uno existente, agregar, buscar, modificar y borrar registros. El nombre del archivo será ingresado por teclado. Cada registro del archivo será un objeto persona con los atributos nombre, dirección y teléfono. Así mismo, para que el usuario pueda elegir cualquiera de las opciones mencionadas, el programa visualizará en pantalla un menú similar al siguiente:

**Archivo actual: ninguno**

**----------**

1. **Nuevo archivo**
2. **Abrir archivo**
3. **Agregar registro**
4. **Buscar un registro**
5. **Buscar siguiente**
6. **Modificar un registro**
7. **Eliminar un registro**
8. **Salir**

**----------**

**Opción (1 - 8): 1**

**Nombre del archivo: telefonos.dat**

La opción *Nuevo* abrirá un archivo para agregar registros; si el archivo existe, preguntará si se desea sobreescribir. La opción *Abrir* permitirá abrir un archivo para leer y escribir o para agregar; estas dos opciones se elegirán de un

menú. La opción *Buscar* permitirá buscar un registro por el campo *nombre*; se permitirá introducir una subcadena de *nombre*, incluso vacía. La opción *Buscar siguiente* buscará el siguiente registro que cumpla con las mismas condiciones que el anteriormente buscado. Finalmente, la opción *Eliminar* permitirá marcar un registro para borrar. Se deberá realizar al menos un método para cada una de las opciones, excepto para las opciones *Buscar*, que compartirán ambas el mismo método, y para *Salir*.

**Nota**: En todos los programas, use excepciones para manejar situaciones anómalas.