Ficheros en C++

Prof. Dr. Nicolás Luis Fernández García

Departamento de Informática y Análisis Numérico
Escuela Politécnica Superior
Universidad de Córdoba

Contenido del tema

1 Ficheros

Contenido de la sección

1. Ficheros
   - Ficheros de cabecera
   - Declaración de flujos
   - Apertura de ficheros
   - Declaración de flujo y apertura de un fichero
   - Existencia de un fichero
   - Cierre de ficheros
   - Lectura y escritura en ficheros de texto
   - Lectura y escritura en ficheros binarios
   - Posicionamiento en un fichero
   - Otras funciones de ficheros

Ficheros

Ficheros de cabecera

Ficheros de cabecera

- #include<iostream>
- #include<fstream>

Contenido de la sección

1. Ficheros
   - Ficheros de cabecera
   - Declaración de flujos
   - Apertura de ficheros
   - Declaración de flujo y apertura de un fichero
   - Existencia de un fichero
   - Cierre de ficheros
   - Lectura y escritura en ficheros de texto
   - Lectura y escritura en ficheros binarios
   - Posicionamiento en un fichero
   - Otras funciones de ficheros

Ficheros

Declaración de flujos

Declaración de flujos

- ifstream entrada;
- ofstream salida;
- fstream entrada_salida;

Contenido de la sección

1 Ficheros
- Ficheros de cabecera
- Declaración de flujos
- Apertura de ficheros
- Declaración de flujo y apertura de un fichero
- Existencia de un fichero
- Cierre de ficheros
- Lectura y escritura en ficheros de texto
- Lectura y escritura en ficheros binarios
- Posicionamiento en un fichero
- Otras funciones de ficheros

Ficheros

Apertura de ficheros

Apertura de un fichero

- void ifstream::open (const char * filename, int ios::openmode mode = ios::in);
- void ofstream::open (const char * filename, int ios::openmode mode= ios::out|ios::trunc);
- void fstream::open (const char * filename, int ios::openmode mode= ios::in|ios::out);

Ficheros

Apertura de ficheros

Modos de apertura de un fichero (1/2)

- ios::in: abre un fichero en modo de entrada (lectura).
  - Si el fichero no existe, falla la apertura
- ios::out: abre un fichero en modo de salida (escritura).
  - Si el fichero existe, lo vacia.
- ios::binary: abre un fichero en modo binario
  - Por defecto, los ficheros son abiertos en modo texto.
  - En modo texto, se puede producir la conversión de algunos caracteres: salto de línea, retorno de carro, etc.
  - Cualquier fichero puede ser abierto en modo texto o en modo binario.

Ficheros

Apertura de ficheros

Modos de apertura de un fichero (2/2)

- ios::trunc: descarta el contenido del fichero si es que existe; es la acción predeterminada de ios::out
- ios::app: la escritura en el fichero siempre se realiza al final.
- ios::ate: abre el fichero y se sitúa al final del fichero.
  - Los datos pueden leerse o escribirse en cualquier parte del fichero.
  - El significado de *ate* es *at end*, al fin.
- ios::nocreate: si no existe el fichero entonces genera un error.
- ios::noreplace: si existe el fichero entoces genera un error

Ficheros

Apertura de ficheros

Combinación de modos de apertura de un fichero

- ios::in|ios::out: abre un fichero para lectura y escritura.
  - Si el fichero no existe, falla la apertura
- ios::in|ios::binary: abre un fichero para lectura en modo binario.
  - Si el fichero no existe, falla la apertura
- ios::in|ios::out|ios::trunc: abre un fichero para lectura y escritura.
  - Si el fichero no existe, lo crea.
  - Si el fichero existe, lo vacia.
- ios::out|ios::app: abre un fichero para añadir.
  - Si el fichero no existe, lo crea.

Ficheros

Apertura de ficheros

Ejemplo (Apertura de ficheros de entrada o lectura)

- *ifstream entrada1, entrada2;*
- *entrada1.open("fichero_entrada1",ios::in);*
- *entrada2.open ("fichero_entrada2");*

Ficheros

Apertura de ficheros

Ejemplo (Apertura de ficheros de salida o escritura)

- *ofstream salida1, salida2;*
- *salida1.open("fichero_salida1",ios::out|ios::trunc);*
- *salida2.open ("fichero_salida2");*

Ficheros

Apertura de ficheros

Ejemplo (Apertura de ficheros de entrada y salida)

- *fstream entrada_salida1, entrada_salida2;*
- *entrada_salida1.open("fichero_salida1",ios::in|ios::out);*
- *entrada_salida2.open("fichero_salida2");*

Contenido de la sección

1. Ficheros
   - Ficheros de cabecera
   - Declaración de flujos
   - Apertura de ficheros
   - Declaración de flujo y apertura de un fichero
   - Existencia de un fichero
   - Cierre de ficheros
   - Lectura y escritura en ficheros de texto
   - Lectura y escritura en ficheros binarios
   - Posicionamiento en un fichero
   - Otras funciones de ficheros

Ficheros

Declaración de flujo y apertura de un fichero

Declaración de flujo y apertura de un fichero

- ifstream entrada (“nombre_fichero”);
- ofstream salida(“nombre_fichero”);
- fstream entrada_salida(“nombre_fichero”);

Contenido de la sección

1. Ficheros
   - Ficheros de cabecera
   - Declaración de flujos
   - Apertura de ficheros
   - Declaración de flujo y apertura de un fichero
   - Existencia de un fichero
   - Cierre de ficheros
   - Lectura y escritura en ficheros de texto
   - Lectura y escritura en ficheros binarios
   - Posicionamiento en un fichero
   - Otras funciones de ficheros

Ficheros

Existencia de un fichero

Existencia de un fichero

- Si la función **open** no puede abrir un fichero entonces el flujo tomará el valor **false**

Ejemplo

```
ifstream entrada;
entrada.open(``fichero_entrada'',ios::in);
if (!entrada)
   cout << ``No se puede abrir el fichero'';
```

Ficheros

Existencia de un fichero

Existencia de un fichero

- Si la función **open** no puede abrir un fichero entonces el flujo tomará el valor **false**

Ejemplo

```
ifstream entrada;
entrada.open(‘‘fichero_entrada’’,ios::in);
if (!entrada)
   cout << ‘‘No se puede abrir el fichero’’;
```

Ficheros

Existencia de un fichero

Existencia de un fichero

- La función **is_open** devuelve **true** si el fichero está abierto y **false** en caso contrario

Ejemplo

```
ifstream entrada;
entrada.open(''fichero_entrada'',ios::in);
if (!entrada.is_open())
   cout << ''No se puede abrir el fichero'';
```

Ficheros

Existencia de un fichero

Existencia de un fichero

- La función **is_open** devuelve **true** si el fichero está abierto y **false** en caso contrario

Ejemplo

```
ifstream entrada;
entrada.open(‘‘fichero_entrada’’,ios::in);
if (!entrada.is_open())
   cout << ‘‘No se puede abrir el fichero’’;
```

Contenido de la sección

1. Ficheros
   - Ficheros de cabecera
   - Declaración de flujos
   - Apertura de ficheros
   - Declaración de flujo y apertura de un fichero
   - Existencia de un fichero
   - Cierre de ficheros
   - Lectura y escritura en ficheros de texto
   - Lectura y escritura en ficheros binarios
   - Posicionamiento en un fichero
   - Otras funciones de ficheros

Ficheros

Cierre de ficheros

Cierre de un fichero

- void **close**();

Ejemplo

- *entrada1.close();*
- *salida1.close();*
- *entrada_salida1.close();*

Ficheros

Cierre de ficheros

Cierre de un fichero

- void **close**();

Ejemplo

- *entrada1.close();*
- *salida1.close();*
- *entrada_salida1.close();*

Contenido de la sección

1. Ficheros
   - Ficheros de cabecera
   - Declaración de flujos
   - Apertura de ficheros
   - Declaración de flujo y apertura de un fichero
   - Existencia de un fichero
   - Cierre de ficheros
   - Lectura y escritura en ficheros de texto
   - Lectura y escritura en ficheros binarios
   - Posicionamiento en un fichero
   - Otras funciones de ficheros

Ficheros

Lectura y escritura en ficheros de texto

Lectura y escritura en ficheros de texto

- Método más sencillo: utilizar los operadores << y >>.

Ficheros

Lectura y escritura en ficheros de texto

Ejemplo (Escritura en un fichero de texto)

```
// Declaracion
ofstream flujo_salida;
// Apertura del fichero
flujo_salida.open(``prueba.txt'');
// Escritura en el fichero
flujo_salida << 10 << `` '' << 19.75 << 'X';
flujo_salida << ``Final del programa'';
// Cierre del fichero
flujo_salida.close();
```

Ficheros

Lectura y escritura en ficheros de texto

Ejemplo (Lectura de un fichero de texto)

```
char caracter; float f; int i; char cadena[80];
// Declaracion
ifstream flujo_entrada;
// Apertura del fichero
flujo_entrada.open(``prueba.txt'');
// Lectura del fichero de texto: entero, real y caracter
flujo_entrada >> i >> f >> caracter;
// Lee hasta que encuentra el primer caracter blanco
flujo_entrada >> cadena;
// Cierre del fichero
flujo_entrada.close();
```

Contenido de la sección

1. Ficheros
   - Ficheros de cabecera
   - Declaración de flujos
   - Apertura de ficheros
   - Declaración de flujo y apertura de un fichero
   - Existencia de un fichero
   - Cierre de ficheros
   - Lectura y escritura en ficheros de texto
   - Lectura y escritura en ficheros binarios
   - Posicionamiento en un fichero
   - Otras funciones de ficheros

Ficheros

Lectura y escritura en ficheros binarios

Lectura de ficheros binarios (1/5)

- **get**:
  - istream & get (char & caracter);
    - Lee un carácter (byte) del flujo de entrada y lo almacena en el parámetro formal *caracter*
  - int get ();
    - Lee un carácter (byte) del flujo de entrada y devuelve su código ASCII
    - Se utiliza para detectar el final de archivo (**eof**) que se suele representar por -1

Ficheros

Lectura y escritura en ficheros binarios

Lectura de ficheros binarios (2/5)

- **get**:
  - istream & get (char *buffer, int numero, char delimitador = '\n');
    - Lee caracteres del flujo de entrada y los almacena en *buffer* hasta que haya leído *numero - 1* caracteres o el carácter delimitador o se haya encontrado el final del fichero.
    - Se añade el carácter nulo al final de *buffer*.
    - Si se encuentra el carácter delimitador, no es extraído, sino que permanece en el flujo de entrada hasta la siguiente operación de lectura.

Ficheros

Lectura y escritura en ficheros binarios

Lectura de ficheros binarios (3/5)

- void **getline**(char *buffer, int numero, char delimitador = '\n');
  - Lee cadenas de caracteres, incluyendo espacios en blanco, y las almacena en *buffer*

Nota

La diferencia entre **get** *y* **getline** *se encuentra en que* **getline** *almacena el carácter delimitador en la cadena antes de añadir el carácter nulo.*

Ficheros

Lectura y escritura en ficheros binarios

Lectura de ficheros binarios (4/5)

- **read**: lee *numero* bytes y los almacena en *buffer*
  - istream & read(char *buffer, int numero);
  - istream & read(unsigned *buffer, int numero);
  - istream & read(signed char *buffer, int numer);

Ficheros

Lectura y escritura en ficheros binarios

Lectura de ficheros binarios (5/5)

- istream & **putback** (char caracter);
  - Devuelve al flujo de entrada el último carácter leído.
- int **peek**():
  - Lee el carácter actual del flujo de entrada pero no avanza.

Ficheros

Lectura y escritura en ficheros binarios

Escritura en ficheros binarios

- ostream & **put** (char caracter);
  - Escribe un carácter (byte) en el flujo de salida
- **write**: escribe *numero* bytes de *buffer* en el flujo de salida
  - ostream & write(const char *buffer, int numero);
  - ostream & write(const unsigned *buffer, int numero);
  - ostream & write(const signed char *buffer, int numero);

Contenido de la sección

1. Ficheros
   - Ficheros de cabecera
   - Declaración de flujos
   - Apertura de ficheros
   - Declaración de flujo y apertura de un fichero
   - Existencia de un fichero
   - Cierre de ficheros
   - Lectura y escritura en ficheros de texto
   - Lectura y escritura en ficheros binarios
   - Posicionamiento en un fichero
   - Otras funciones de ficheros

Ficheros

Posicionamiento en un fichero

Posicionamiento en un fichero de lectura

- **seekg**:
  - istream & seekg(streampos pos);
    - Posiciona el cursor de lectura en el lugar indicado por *pos*
  - istream & seekg(streamoff desp, seek_dir dir);
    - Desplaza el cursor de lectura el número de bytes indicados por *desp* teniendo en cuenta la dirección especificada por *dir*: *beg*, *cur*, *end* (principio, posición actual y final del fichero).
- streampos **tellg**();
  - Devuelve la posición actual del flujo de entrada o -1 si se produce un error.

Ficheros

Posicionamiento en un fichero

Posicionamiento en un fichero de escritura

- **seekp**:
  - ostream & seekp(streampos pos);
    - Posiciona el cursor de escritura en el lugar indicado por *pos*
  - ostream & seekp(streamoff desp, seek_dir dir);
    - Desplaza el cursor de escritura el número de bytes indicados por *desp* teniendo en cuenta la dirección especificada por *dir*: *beg, cur, end* (principio, posición actual y final del fichero).
- streampos **tellp**();
  - Devuelve la posición actual del flujo de salida o -1 si se produce un error.

Contenido de la sección

1. Ficheros
   - Ficheros de cabecera
   - Declaración de flujos
   - Apertura de ficheros
   - Declaración de flujo y apertura de un fichero
   - Existencia de un fichero
   - Cierre de ficheros
   - Lectura y escritura en ficheros de texto
   - Lectura y escritura en ficheros binarios
   - Posicionamiento en un fichero
   - Otras funciones de ficheros

Ficheros

Otras funciones de ficheros

Otras funciones

- bool **eof**() const;
  - Devuelve **true** si se ha alcanzado el final del archivo; en caso contrario, devuelve **false**
- bool **good**() const;
  - Indica si la operación de lectura anterior ha tenido éxito.
- bool **fail**() const;
  - Indica que la siguiente operación de lectura fallará.
- bool **bad**() const;
  - El flujo de entrada está corrompido.

Ficheros en C++

Prof. Dr. Nicolás Luis Fernández García

Departamento de Informática y Análisis Numérico
Escuela Politécnica Superior
Universidad de Córdoba