Manual de la Biblioteca IOCONSOLE
DRAFT

1 Instalación de la Biblioteca IOCONSOLE

1.1 Linux y MacOS-X

1. Para saber la versión del compilador GNU G++ introduzca el siguiente comando desde un terminal:

   ```
   $ g++ --version
   ```

   También el siguiente comando mostrará las versiones instaladas:

   ```
   $ ls -l /usr/include/c++
   ```

2. Comprobar si la biblioteca ya está instalada. Introduzca el siguiente comando:

   ```
   $ ls -l /usr/include/c++/*/ext/ioconsole.hpp
   ```

   En caso de no estar previamente instalada, el comando anterior mostrará algo similar a:

   ```
   ls: no se puede acceder a /usr/include/c++/*/ext/ioconsole.hpp: No existe el fichero o el directorio
   ```

   Si la biblioteca ya se encuentra instalada, mostrará algo similar a: (donde <*version*> es la versión de su compilador GNU G++)

   ```
   -rw-r--r-- 1 root root 2143 dic 31  2008 /usr/include/c++/<version>/ext/ioconsole.hpp
   ```

3. Si la biblioteca no está previamente instalada, continúe con los siguientes pasos.

4. Crear el directorio `ext` con el siguiente comando: (donde <*version*> es la versión de su compilador GNU G++)

   ```
   $ sudo mkdir /usr/include/c++/<version>/ext
   ```

5. Copiar el archivo `ioconsole.hpp` que se adjunta al directorio anterior:

   ```
   $ sudo cp ioconsole.hpp /usr/include/c++/<version>/ext
   ```

6. Comprobar que el archivo `ioconsole.hpp` se ha copiado al directorio adecuado

   ```
   $ ls -l /usr/include/c++/*/ext/ioconsole.hpp
   ```

1.2 Windows CodeBlocks C++

1. Comprobar si la carpeta `ext` se encuentra en la siguiente carpeta: (donde <*version*> es la versión de su compilador GNU G++)

   ```
   \Equipo\C:\Archivos de Programa\CodeBlocks\MinGW\lib\gcc\mingw32\<version>\include\c++\
   ```

2. Si la carpeta `ext` no existe, entonces crearla en la carpeta especificada anteriormente (paso 1).

3. Copiar el archivo `ioconsole.hpp` que se adjunta a la carpeta creada anteriormente (en caso de que ya exista, reemplazarlo):

   ```
   \Equipo\C:\Archivos de Programa\CodeBlocks\MinGW\lib\gcc\mingw32\<version>\include\c++\ext\
   ```

4. Comprobar que el archivo `ioconsole.hpp` se ha copiado a la carpeta especificada anteriormente (paso 3).

1.3 En Directorio o Carpeta de Trabajo

1. Copiar el archivo `ioconsole.hpp` que se adjunta al directorio o carpeta de trabajo.

2 Utilización de la Biblioteca IOCONSOLE

La biblioteca *ioconsole* proporciona un mecanismo adecuado para manipular de forma flexible la salida a consola, así como un mecanismo para realizar la entrada de datos *sin buffer y sin eco*. Es adecuada para desarrollar programas interactivos que requieran realizar la entrada/salida de información textual de una manera flexible, y controlada por el propio programa. Así mismo, también proporciona suprogramas para generar números aleatorios, y para dormir el proceso un tiempo especificado. En su conjunto, la biblioteca *ioconsole* proporciona un entorno adecuado para desarrollar aplicaciones textuales interactivas, así como simples juegos interactivos con una interfaz textual.

Una vez instalada la biblioteca, para poder utilizarla nuestro programa C++ debe incluir el siguiente archivo de cabecera:

```
#include <ext/ioconsole.hpp>
```

En caso de que la biblioteca se haya instalado en el directorio de trabajo, nuestro programa C++ incluirá la biblioteca de la siguiente manera:

```
#include "ioconsole.hpp"
```

Para facilitar la utilización de la biblioteca, por defecto las operaciones de salida de datos invocan automáticamente a `flush` para forzar el volcado de la salida a pantalla de forma instantánea. Sin embargo, este hecho puede producir un efecto adverso de *parpadeo* en determinadas aplicaciones y juegos interactivos. Por ello, si el programador desea evitar el volcado automático de la salida a pantalla, deberá definir el símbolo `NAFLUSH__` antes de incluir la biblioteca.

```
#define NAFLUSH__
#include <ext/ioconsole.hpp>
```

En este modo de operación, el programador deberá invocar explícitamente el volcado de la salida (`flush`) cuando lo considere adecuado.

Esta biblioteca se encuentra definida dentro del espacio de nombres `ioconsole`, sin embargo, para facilitar su utilización dicho espacio de nombres se encuentra expandido explícitamente.

La *entidad* `cio` nos permite manipular la entrada/salida directa a consola.

2.1 Miscelánea y Números Aleatorios

- Borra la pantalla

  ```
  void clear();
  ```

- Pausa hasta pulsar la tecla ENTER

  ```
  void pausa();
  ```

- Duerme el proceso el número de milisegundos especificado

  ```
  void msleep(unsigned milliseconds);
  ```

- Devuelve un número aleatorio entre [0..max) (exclusive)

  ```
  unsigned aleatorio(unsigned max);
  ```

- Devuelve un número aleatorio entre [inf..sup] ambos inclusive

  ```
  unsigned aleatorio(unsigned inf, unsigned sup);
  ```

2.2 Información de Consola

La estructura `text_info` nos permite obtener información sobre la consola:

```
struct text_info {
    unsigned char winleft;      // Coordenada de la columna izquierda
    unsigned char wintop;       // Coordenada de la fila superior
    unsigned char winright;     // Coordenada de la columna derecha
    unsigned char winbottom;    // Coordenada de la fila inferior
    unsigned char attribute;    // Atributo del texto
    unsigned char normattr;     // Atributo normal
    unsigned char currmode;     // Modo de video actual
    unsigned char screenheight; // Numero de lineas de la pantalla
    unsigned char screenwidth;  // Numero de columnas de la pantalla
    unsigned char curx;         // Columna donde se encuentra el cursor
    unsigned char cury;         // Fila donde se encuentra el cursor
};
```

Podemos extraer esta información de la consola declarando una variable de tipo `text_info`, y realizando una lectura desde `cio` sobre ella:

```
text_info info;  // declaracion de variable 'info' de tipo 'text_info'
cio >> info;
```

Para saber la posición actual del cursor en la consola (la esquina superior izquierda de la consola se corresponde con las coordenadas [1,1]):

```
unsigned x, y;
cio >> wherexy(x, y);
```

2.3 Colores

```
BLACK, BLUE, GREEN, CYAN, RED, MAGENTA, BROWN, LIGHTGRAY,

DARKGRAY, LIGHTBLUE, LIGHTGREEN, LIGHTCYAN, LIGHTRED, LIGHTMAGENTA, YELLOW, WHITE
```

2.4 Salida a Consola

Las siguientes opciones para salida de datos a consola se pueden combinar como sean necesarias.

- Para mostrar datos de los tipos básicos: `char`, `short`, `ushort`, `int`, `uint`, `long`, `ulong`, `float`, `double`, `long double`, `string`:

  ```
  cio << x ;
  ```

- Para poner la posición actual del cursor en la consola y mostrar un dato (la esquina superior izquierda de la consola se corresponde con las coordenadas [1,1]):

  ```
  cio << cursorxy(3, 5) << x ;
  ```

- Para ocultar el cursor:

  ```
  cio << no_cursor ;
  ```

- Para mostrar el cursor (normal):

  ```
  cio << normal_cursor ;
  ```

- Para mostrar el cursor (solid):

  ```
  cio << solid_cursor ;
  ```

- Para borrar la pantalla:

  ```
  cio << clear_screen ;
  ```

- Para borrar la linea desde la posición del cursor hasta el final:

  ```
  cio << clear_line ;
  ```

- Para eliminar la linea donde se encuentra el cursor:

  ```
  cio << delete_line ;
  ```

- Para insertar una nueva linea donde se encuentra el cursor:

  ```
  cio << insert_line ;
  ```

- Para mostrar un dato con un determinado color de texto (ver códigos de colores en sección 2.3):

  ```
  cio << color(RED) << x << normal_video ;
  ```

- Para mostrar un dato con un determinado color de texto y de fondo (ver códigos de colores en sección 2.3):

  ```
  cio << color(RED, CYAN) << x << normal_video ;
  ```

- Para mostrar un dato con un determinado color de fondo (ver códigos de colores en sección 2.3):

  ```
  cio << bgcolor(CYAN) << x << normal_video ;
  ```

- Para mostrar un dato con video intenso:

  ```
  cio << intense_video << x << normal_video ;
  ```

- Para mostrar un dato con video intenso:

  ```
  cio << high_video << x << normal_video ;
  ```

- Para mostrar un dato con video de baja intensidad:

  ```
  cio << low_video << x << normal_video ;
  ```

- Para mostrar un dato con parpadeo:

  ```
  cio << blink_video << x << no_blink_video ;
  ```

- Para mostrar un dato con una anchura de campo:

  ```
  cio << setw(5) << x ;
  ```

- Para mostrar un dato con una anchura de campo, especificando el carácter de relleno:

  ```
  cio << setfill('0') << setw(5) << x ;
  ```

- Para mostrar un dato real especificando la precisión:

  ```
  cio << setprecision(3) << x ;
  ```

- Para mostrar un dato en representación decimal:

  ```
  cio << dec << x ;
  ```

- Para mostrar un dato en representación hexadecimal:

  ```
  cio << hex << x ;
  ```

- Para mostrar un dato en representación octal:

  ```
  cio << oct << x ;
  ```

- Para mostrar un dato lógico (`bool`) en representación textual:

  ```
  cio << boolalpha << x ;
  ```

- Para forzar que el contenido del *buffer de salida* se muestre por pantalla:

  ```
  cio << flush ;
  cio.flush() ;
  ```

- Para enviar un salto de línea a consola (también hace flush):

  ```
  cio << endl ;
  cio.endl() ;
  ```

2.5 Teclas Especiales

```
KEY_ENTER, KEY_BACKSP, KEY_ESC, KEY_TAB, KEY_SHIFT_TAB,

KEY_UP,   KEY_DOWN, KEY_LEFT, KEY_RIGHT,  KEY_INSERT,
KEY_HOME, KEY_END,  KEY_PGUP, KEY_PGDOWN, KEY_DELETE,

KEY_SHIFT_UP,   KEY_SHIFT_DOWN, KEY_SHIFT_LEFT, KEY_SHIFT_RIGHT,  KEY_SHIFT_INSERT,
KEY_SHIFT_HOME, KEY_SHIFT_END,  KEY_SHIFT_PGUP, KEY_SHIFT_PGDOWN, KEY_SHIFT_DELETE,

KEY_META_UP,   KEY_META_DOWN, KEY_META_LEFT, KEY_META_RIGHT,  KEY_META_INSERT,
KEY_META_HOME, KEY_META_END,  KEY_META_PGUP, KEY_META_PGDOWN, KEY_META_DELETE,

KEY_CTRL_UP,   KEY_CTRL_DOWN, KEY_CTRL_LEFT, KEY_CTRL_RIGHT,  KEY_CTRL_INSERT,
KEY_CTRL_HOME, KEY_CTRL_END,  KEY_CTRL_PGUP, KEY_CTRL_PGDOWN, KEY_CTRL_DELETE,

KEY_SHIFT_META_UP,   KEY_SHIFT_META_DOWN, KEY_SHIFT_META_LEFT, KEY_SHIFT_META_RIGHT,  KEY_SHIFT_META_INSERT,
KEY_SHIFT_META_HOME, KEY_SHIFT_META_END,  KEY_SHIFT_META_PGUP, KEY_SHIFT_META_PGDOWN, KEY_SHIFT_META_DELETE,

KEY_SHIFT_CTRL_UP,   KEY_SHIFT_CTRL_DOWN, KEY_SHIFT_CTRL_LEFT, KEY_SHIFT_CTRL_RIGHT,  KEY_SHIFT_CTRL_INSERT,
KEY_SHIFT_CTRL_HOME, KEY_SHIFT_CTRL_END,  KEY_SHIFT_CTRL_PGUP, KEY_SHIFT_CTRL_PGDOWN, KEY_SHIFT_CTRL_DELETE,
```

2.6 Entrada de Consola

- Para leer un dato de los tipos básicos (salta espacios iniciales): `char`, `short`, `ushort`, `int`, `uint`, `long`, `ulong`, `float`, `double`, `long double`, `string`:

```
cio >> x ;
```

- Para leer una linea (no salta espacios iniciales):

```
string str ;
getline(cio, str) ;
```

- Para leer un carácter (no salta espacios iniciales):

```
char c ;
cio.get(c) ;
```

- Elimina los espacios iniciales del buffer de entrada (para utilizar antes de `cio.get(c)` y `getline(cio, str)` en caso de ser necesario):

```
cio >> ws ; getline(cio, str) ;
cio >> ws ; cio.get(c) ;
```

- Elimina los caracteres del buffer de entrada (para utilizar después de leer un dato con `cio >> x` en caso de ser necesario):

```
cio >> x >> ignore ;
```

- Devuelve el carácter al buffer de entrada

```
cio.ungetch(c) ;
```

- Deshabilita el *buffer de entrada*. Debe deshabilitarse **antes** de realizar alguna entrada de datos.

```
cio >> disable_buffer ;
cio.disable_input_buffer() ;
```

- Lee una tecla (sin eco, sin buffer) (ver códigos de teclas especiales en sección 2.5):

```
unsigned k ;
cio >> key(k) ;
```

- Para saber si se ha pulsado una tecla (ver códigos de teclas especiales en sección 2.5):

```
if (cio.kbhit()) {
    unsigned k ;
    cio >> key(k) ;
    ...
}
```

2.7 Control del Estado de la Consola

- Para saber si la consola se encuentra en un estado adecuado:

```
if (cio.fail()) {
    ...
}
```

- Para restaurar la consola a un estado adecuado:

```
if (cio.fail()) {
    cio.clear() ;
    ...
}
```

- Para acceder y manipular los *flags* que influyen en el comportamiento de *cio*, véase los correspondientes a `cout` de `<iostream>`.

```
cio.flags( ... ) ;
cio.setf( ... ) ;
cio.unsetf( ... ) ;
```

3 Biblioteca CONIO

La biblioteca `<ext/ioconsole.hpp>`, además de la interfaz explicada anteriormente, también proporciona el interfaz de la biblioteca `<conio.h>`. Es la encargada de manipular directamente tanto la salida a consola como la entrada desde el teclado. Esta biblioteca no forma parte del estándar ANSI, por lo que hará que nuestros programas no sean portables. Se debe incluir el siguiente fichero de cabecera:

```
#include <ext/ioconsole.hpp>
```

Los siguientes tipos y funciones se encuentran definidos dentro del espacio de nombres `conio`:

```
struct text_info {
    unsigned char winleft;      // Coordenada de la columna izquierda
    unsigned char wintop;       // Coordenada de la fila superior
    unsigned char winright;     // Coordenada de la columna derecha
    unsigned char winbottom;    // Coordenada de la fila inferior
    unsigned char attribute;    // Atributo del texto
    unsigned char normattr;     // Atributo normal
    unsigned char currmode;     // Modo de video actual
    unsigned char screenheight; // Numero de lineas de la pantalla
    unsigned char screenwidth;  // Numero de columnas de la pantalla
    unsigned char curx;         // Columna donde se encuentra el cursor
    unsigned char cury;         // Fila donde se encuentra el cursor
};
```

La esquina superior izquierda de la consola se corresponde con las coordenadas [1,1].

3.1 Información

- Obtener información sobre la ventana actual

  ```
  void gettextinfo(struct text_info* _r);
  ```

- Obtener el valor de la coordenada "x" donde se encuentra el cursor actualmente

  ```
  int wherex();
  ```

◦ Obtener información sobre la ventana actual (C++)

```
void gettextinfo(struct text_info& _r);
```

- Obtener el valor de la coordenada "y" donde se encuentra el cursor actualmente

  ```
  int wherey();
  ```

3.2 Salida y entrada de texto

- Envía texto formateado a la consola

  ```
  int cprintf(const char fmt[], ...);
  ```

- Envía una cadena de caracteres a consola

  ```
  int cputs(const char _str[]);
  ```

◦ Envía un string a consola (C++)

```
void cputs(const string& _str);
```

- Envía un carácter simple a consola

  ```
  int putch(int _c);
  ```

- Entrada formateada de datos de consola

  ```
  int cscanf(const char fmt[], ...);
  ```

- Entrada de una cadena de caracteres de consola, donde `_str[0]` indica el máximo número de caracteres a leer (< 256). Al término de la función, `_str[1]` contiene el número de caracteres realmente leídos (`_str` debe tener un tamaño de `_str[0]+2`). La función devuelve el puntero al primer carácter leído.

  ```
  char *cgets(char _str[]);
  ```

◦ Entrada de un string de consola (C++)

```
void cgets(string& _str);
```

- Comprueba si hay pulsaciones de teclado disponibles para entrada

  ```
  int kbhit();
  ```

- Lee un carácter directamente de consola (sin eco, sin buffer). En caso de *tecla especial*, la primera llamada devolverá el carácter `'\0'`, y una segunda llamada producirá el carácter correspondiente a la tecla especial.

  ```
  int getch();
  ```

◦ Lee un carácter directamente de consola (sin eco, sin buffer) En caso de *tecla especial*, véase códigos de teclas especiales en la sección 2.5. (C++)

```
int getkey();
```

- Lee un carácter directamente de consola (con eco, sin buffer)

  ```
  int getche();
  ```

- Devuelve el carácter al buffer de entrada

  ```
  int ungetch(int _c);
  ```

◦ Deshabilita el *buffer de entrada*. Debe deshabilitarse **antes** de realizar alguna entrada de datos.

```
void disable_input_buffer();
```

3.3 Manipulacion del cursor en consola

- Posiciona el cursor directamente en una coordenada de consola (esquina superior izquierda [1,1])

```
void gotoxy(int x, int y);
```

- Borra e inicializa toda la consola

```
void clrscr();
```

- Borra la linea actual desde el cursor hasta el final

```
void clreol();
```

- Especifica el tipo de cursor: `_NOCURSOR`, `_SOLIDCURSOR`, `_NORMALCURSOR`

```
void _setcursortype(int _type);
```

- Elimina la linea actual donde se encuentra el cursor

```
void delline();
```

- Inserta una linea nueva en blanco en la linea actual

```
void insline();
```

3.4 Ajuste de atributos del texto (intensidad, color del texto y del fondo)

- Actualiza el color del texto que se mostrará a continuación

```
void textcolor(int _color);
```

- Actualiza el color de fondo del texto que se mostrará a continuación

```
void textbackground(int _color);
```

- Actualiza el color del texto y de fondo que se mostrará a continuación

```
void textattr(int _attr);
```

- Actualiza la intensidad del texto que se mostrará a continuación (Alta intensidad)

```
void intensevideo();
```

- Actualiza la intensidad del texto que se mostrará a continuación (Alta intensidad)

```
void highvideo();
```

- Actualiza la intensidad del texto que se mostrará a continuación (Baja intensidad)

```
void lowvideo();
```

- Actualiza la intensidad del texto que se mostrará a continuación (intensidad normal)

```
void normvideo();
```

- Actualiza la intensidad del texto que se mostrará a continuación (Parpadeo)

```
void blinkvideo();
```

3.5 Control de ventanas (NO IMPLEMENTADO)

- Pone la consola en modo de texto (NO IMPLEMENTADO)

```
void textmode(int _mode);
```

- Define una ventana de texto (NO IMPLEMENTADO)

```
void window(int _left, int _top, int _right, int _bottom);
```

3.6 Movimientos de bloques (NO IMPLEMENTADO)

- Copia texto de una zona de pantalla a otra (NO IMPLEMENTADO)

```
int movetext(int _left, int _top, int _right, int _bottom, int _destleft, int _desttop);
```

- Copia texto de pantalla a memoria (NO IMPLEMENTADO)

```
int gettext(int _left, int _top, int _right, int _bottom, void*_destin);
```

- Copia texto de memoria a pantalla (NO IMPLEMENTADO)

```
int puttext(int _left, int _top, int _right, int _bottom, void*_source);
```