ENTRADA Y SALIDA DE DATOS

By: Candy

ACLARACIÓN

A diferencia de ‘C’ en ‘C++’, y más específicamente en el curso, los controles de entrada y salida de datos ya no son manejados por funciones, sino directamente por “OBJETOS”. Estos se encuentran en las librerías iostream, iomanip, fstream (está es para archivos).

Para esta parte del curso usaremos ‘2’ de esos objetos.

Candy

Objeto ‘cout’

Usaremos este objeto para enviar los datos al medio estándar de salida; es decir, para escribir.

FORMATO GENERAL

```
// Formato de uso: cout<<{valor};
cout<<"texto"<<endl;
cout<<15<<endl;
cout<<'a'<<endl;
cout<<2.76<<endl;
bool var = true;
cout<<(var and false)<<endl;
```

Se debe poner “cout” seguido del operador “<<”, para finalmente poner el valor deseado a mostrar en consola.

SALIDA

Nota: Las operaciones por operadores “<<” y “>>” son agrupables en la misma línea ilimitadamente como se muestra en el ejemplo. (Solo para estos casos cuando trabajamos con objetos de la librería iostream y fstream)

Candy

MÉTODOS IMPORTANTES

```
cout.width(int n);
cout.fill(char c)
cout<<left;
```

```
cout<<setw(int n);
cout<<setfill(char c);
cout<<right;
```

Imagina que pones una celda fija de cierto tamaño de caracteres (**width**), para luego poner tu valor como texto ahí dentro alineado de cierta manera (**right** o **left**).
Si es que sobra espacio en tu celda se rellenará con caracteres (**fill**).

```
cout.precision(int n);
cout<<setprecision(int n);
cout<<fixed;
```

A veces tendremos que controlar el número de decimales que se muestran en los resultados. Para esto el método **precision**, pues por si solo recorta los números impresos hacia la cantidad indicada. No obstante, si solo queremos recortar los decimales, debemos agregar el **fixed**.

Candy

Ahora veremos un uso continuo de los métodos..

De manera rápida se diría que tenemos la impresión de un valor en una celda de tamaño '10' para luego alinearlo a la izquierda nuevamente en una celda de tamaño '10'. Luego, cambiamos el carácter de rellenado a la letra 'A' , lo alineamos a la derecha, y volvemos a imprimir el valor en una celda de tamaño '10'. Después volvemos a cambiar el carácter de rellenado a la letra 'B' y lo alineamos a la izquierda en una celda de tamaño '10'. Luego imprimimos el valor decimal con una precisión de 5 caracteres literalmente (sin incluir el punto). Finalmente imprimimos el valor decimal ahora con una precisión de 5 decimales.

```
int num = 1234567;
double dec = 242.13278;
cout.width(10);
cout<<num<<"'"<<endl;
cout<<left<<setw(10)<<num<<"'"<<endl;
cout.fill('A');
cout<<right;
cout.width(10);
cout<<num<<"'"<<endl;
cout<<setfill('B')<<left<<setw(10)<<num<<"'"<<endl;
cout.precision(5);
cout<<right<<dec<<endl;
cout<<fixed<<dec<<endl;
```

```
   1234567'
1234567   '
AAA1234567'
1234567BBB'
242.13
242.13278

RUN SUCCESSFUL (total time: 43ms)
```

Notas:
- Por default la alineación en una celda es a la derecha.
- Por default el rellenado es con espacios en blanco.
- Los métodos que controlan el 'width' solo afectan a la próxima impresión, a diferencia del resto de métodos que sus cambios sí se mantienen hasta que se vuelva a cambiar.
- Si no se usa 'width', simplemente se imprimirá normalmente sin importar la alineación, pues no existe 'celda'.

Candy

Objeto ‘cin’

Usaremos este objeto para leer los datos desde el medio estándar de entrada.

FORMATO GENERAL

```
int num = 0;
double dec = 0.57;
char car = 'a';

// Formato de uso: cin>>{variable}
cout<<"num = "<<num<<endl<<"Ingrese el nuevo 'num': ";
cin>>num;
cout<<"num = "<<num<<endl;

cout<<"dec = "<<dec<<endl<<"Ingrese el nuevo 'dec': ";
cin>>dec;
cout<<"dec = "<<dec<<endl;

cout<<"car = "<<car<<endl<<"Ingrese el nuevo 'car': ";
cin>>car;
cout<<"car = "<<car<<endl;
```

Se debe poner “cin” seguido del operador “>>”, para finalmente poner la variable en donde se almacenará el valor deseado a leer por consola.

ACLARACIÓN

No se puede ingresar datos por la consola interna de netbeans!!

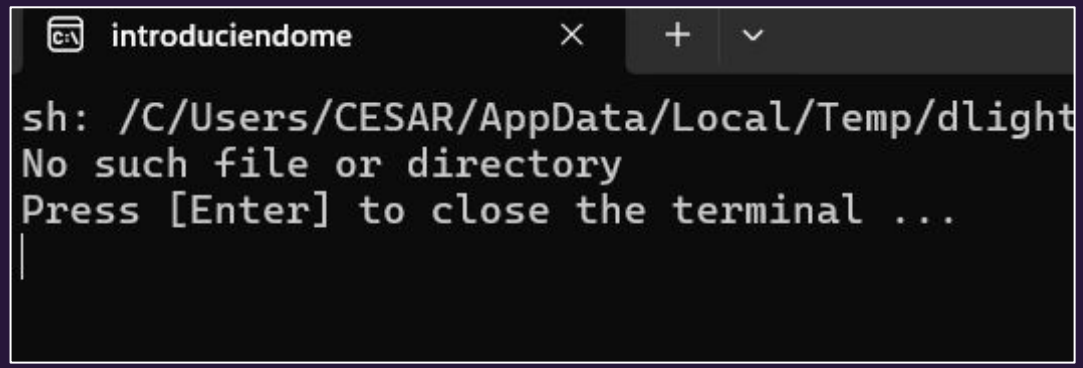
y ahora? ..

Candy

CONSOLA EXTERNA

Dado que no tenemos forma de enviar datos por la consola interna de netbeans, debemos lograrlo de alguna otra manera. Para esto usaremos la consola externa.. (o terminal???)

No obstante, si hacemos el método dentro de netbeans, existe la posibilidad de recibir un error, aunque se haga reiteradas veces, que hace que no se encuentre un directorio para la terminal externa..



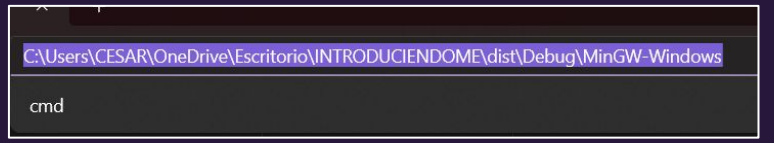
Por eso realizaremos la forma manual que es la que recomiendo para sus laboratorios!

Candy

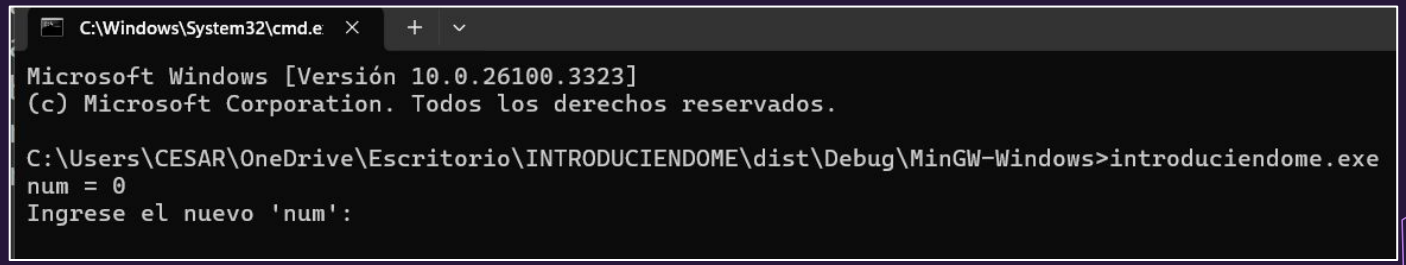
CONSOLA EXTERNA

Nos iremos a la carpeta de nuestro proyecto y entraremos en la carpeta **dist**, así hasta entramos en todas las subcarpetas hasta llegar al ejecutable de nuestro programa.

Luego le daremos click en la barra de arriba con la dirección de nuestro directorio y escribiremos “cmd”, para acceder a la terminal en este directorio.

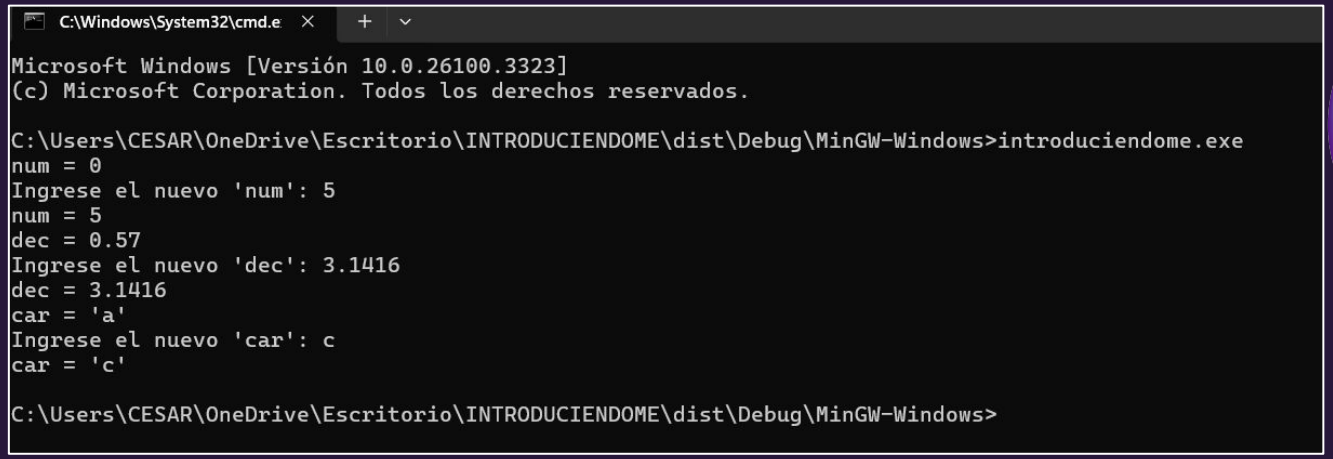


Luego damos “tab” o escribimos directamente el nombre de nuestro ejecutable con su respectiva extensión “.exe” y lo ejecutamos dando “enter”. Ahora sí podremos ingresar los datos.



Candy

Así resultaría nuestro programa a partir del código mostrado.



De manera rápida explicando el código, simplemente actualizamos el valor de cada variable, por otro ingresado por el usuario. Cabe resaltar que se esperan datos correctos, pues una incorrecta lectura puede llevar a errores y excepciones.

Nota:
No es necesario que cierren la terminal cada vez que quieran actualizar el programa. Simplemente, luego de compilarlo en netbeans, vuelvanlo a ejecutar en la misma terminal y ya estará actualizado.

Candy

MÉTODOS IMPORTANTES

Cuando ingresas datos al computador, básicamente ahí creas tu buffer de entrada. Pues no es necesario ingresar por separado cada lectura, puedes ingresar todo de golpe, y el programa irá leyendo y usando en orden lo recibido previamente hasta terminarse el “buff”.
Teniendo esta idea, y para resumir, debemos pensarlo como si fuésemos el cursor dentro de un archivo de texto.

cin.get();

Con el método get, devolveremos el próximo carácter y moveremos una posición el cursor.

cin.unget();

Con el método unget, simplemente retrocedemos una posición el cursor.

cin.peek();

Con el método peek, únicamente devolveremos el próximo carácter.

cin>>ws;

Cabe resaltar esta función también. Simplemente te mueve el cursor hasta chocar con un carácter que no sea considerado “whitespace”; es decir, que no sea espacio, enter, o tabs.

cin.eof();

Esta función solo la usaremos cuando trabajemos con archivos o buffers de los cuales no tendremos una longitud fija o predeterminada. Este devuelve verdadero o falso dependiendo si se alcanzó el final de los datos.

Candy

Ahora veremos un uso continuo de los métodos..
Se ingresará el buffer “ a1b 2c3”

El primer get nos obtiene un espacio en blanco, pues el primer caracter del buffer sería este mismo.
De igual manera obtenemos un espacio en blanco con el segundo get, pues el segundo carácter del buff sigue siendo eso. Ahora hacemos uso de la función ws para pasar directamente detrás de la ‘a’.
Luego obtenemos a la ‘a’ antes mencionada con el siguiente get.
Ahora hacemos unget, por lo que retrocedemos lo que habíamos avanzado al hacer el ultimo get, siendo entonces nuestro próximo carácter la letra ‘a’ en lugar del ‘1’.
Entonces ahora, al hacer get, obtenemos otra vez el carácter ‘a’.
Luego hacemos peek, por lo que solo copiamos el próximo carácter sin movernos, obteniendo así el carácter ‘1’.
Finalmente volveremos a hacer get, y obtenemos nuevamente el carácter ‘1’, pues no nos movimos por el peek.

```
cout<<"Ingresame el real buffer: ";
// Aquí ingresaremos el buff "      a1b 2c3"
char c = cin.get();
cout<<"Caracter obtenido: '"<<c<<"'"<<endl;
c = cin.get();
cout<<"Caracter obtenido: '"<<c<<"'"<<endl;
cin>>ws;
c = cin.get();
cout<<"Caracter obtenido: '"<<c<<"'"<<endl;
cin.unget();
c = cin.get();
cout<<"Caracter obtenido: '"<<c<<"'"<<endl;
c = cin.peek();
cout<<"Caracter obtenido: '"<<c<<"'"<<endl;
c = cin.get();
cout<<"Caracter obtenido: '"<<c<<"'"<<endl;
```

```
C:\Users\CESAR\OneDrive\Escritorio\INTRO
Ingresame el real buffer:      a1b 2c3
Caracter obtenido: ' '
Caracter obtenido: ' '
Caracter obtenido: 'a'
Caracter obtenido: 'a'
Caracter obtenido: '1'
Caracter obtenido: '1'
```

Nota:
No es necesario almacenar el carácter devuelto por el uso de get, si es que no lo necesitan, y tan solo quieren moverse un carácter, solo pongan la función y listo.

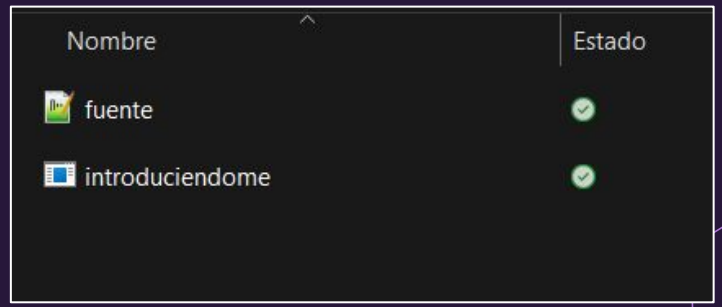
Candy

REDIRECCIONAMIENTO DE ENTRADA Y SALIDA DE DATOS

Para resumir el tema, es simplemente utilizar archivos que sirvan como fuente y destino, en lugar de hacerlo solo por consola. Para redireccionar solo hay que agregar algo a nuestra línea de comandos cuando ejecutamos el programa.

Redirección de salida: Proyecto.exe> Salida.ext
Redirección de entrada: Proyecto.exe< Entrada.ext
Redirección de entrada y salida: Proyecto.exe< Entrada.ext> Salida.ext
Donde: ".ext" sería la extensión del archivo, usualmente ".txt" para esta parte del curso.

Nota:
Dado que cuando redireccionamos, debemos hacerlo hacia o desde rutas específicas, para hacerlo de manera sencilla, solo trabajaremos con archivos dentro del mismo directorio que el ejecutable.



Candy

REDIRECCIONAMIENTO DE ENTRADA Y SALIDA DE DATOS

1) Primero redireccionemos la salida, para esto creamos un programa cualquiera, por ejemplo este que imprime los números del 1 al 10.

```
int main(int argc, char** argv) {

    for(int i = 0;i < 10;i++) cout<<i+1<<endl;


    return 0;
}
```

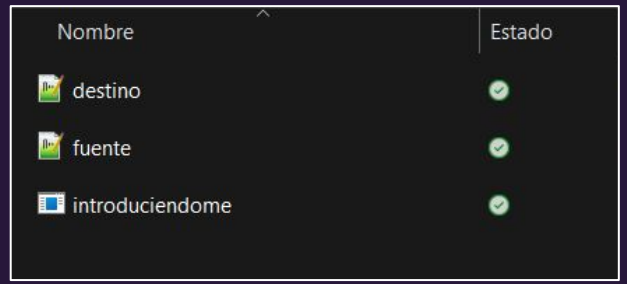
2) Luego escribimos la redirección en la línea de comando utilizando nuestros nombres de archivo y damos enter.

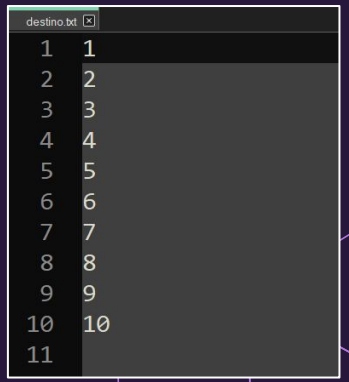
```
ug\MinGW-Windows>introduciendome.exe > destino.txt

ug\MinGW-Windows>
```

Nota:
No es necesario que exista el archivo “destino”, pues en caso de no haber el mencionado, automáticamente se creará uno, con el contenido correspondiente.

3) Luego notaremos que el archivo propuesto tendrá el contenido deseado.





Candy

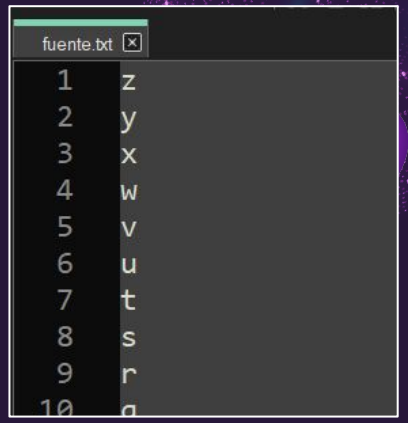
REDIRECCIONAMIENTO DE ENTRADA Y SALIDA DE DATOS

1) Ahora redireccionemos solo la entrada, y mostraremos en consola. Para esto creamos un programa cualquiera, por ejemplo este que imprimirá el contenido recibido desde el usuario..

```
int main(int argc, char** argv) {
    char letra;
    cout<<"LETRAS"<<endl;
    for(int i = 0;1;i++) {
        cin>>letra;
        if(cin.eof()) break;
        cout<<letra;
    }

    return 0;
}
```

2) Luego escribimos la redirección en la línea de comando utilizando nuestros nombres de archivo y damos enter.




```
GW-Windows>introduciendome.exe< fuente.txt
```

3) Luego notaremos la salida en consola obtenida.

```
C:\Users\CESAR\OneDrive\Escritor
LETRAS
zyxwvutsrqponmlkjihgfedcba
```

Candy

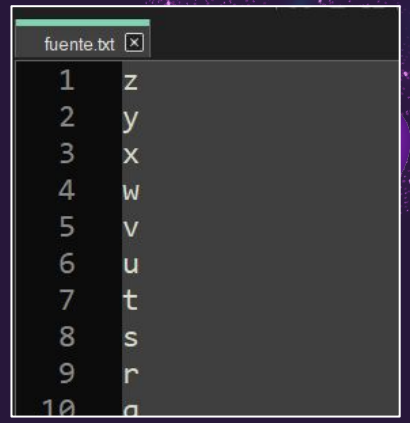
REDIRECCIONAMIENTO DE ENTRADA Y SALIDA DE DATOS

1) Finalmente ahora probaremos ambas redirecciones, para esto usaremos el programa anterior, y lo enviaremos al archivo destino.

```
int main(int argc, char** argv) {
    char letra;
    cout<<"LETRAS"<<endl;
    for(int i = 0;1;i++) {
        cin>>letra;
        if(cin.eof()) break;
        cout<<letra;
    }

    return 0;
}
```

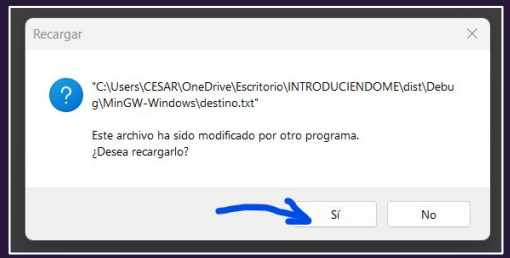
2) Luego escribimos la redirección en la línea de comando utilizando nuestros nombres de archivo y damos enter.

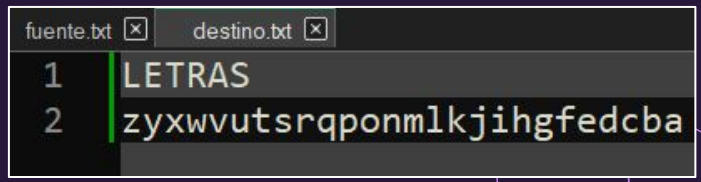



```
GW-Windows>introduciendome.exe< fuente.txt> destino.txt

GW-Windows>
```

3) Luego notaremos que el archivo propuesto nos dirá que debe recargarse (aceptamos la misma), o directamente tendrá el contenido deseado.





Nota:
Esto fue realizado con Notepad++, en caso no te aparezca lo de recargar, simplemente vuelve a abrir el archivo.

Candy

GRACIAS!

By: Candy