ARCHIVO DE DATOS

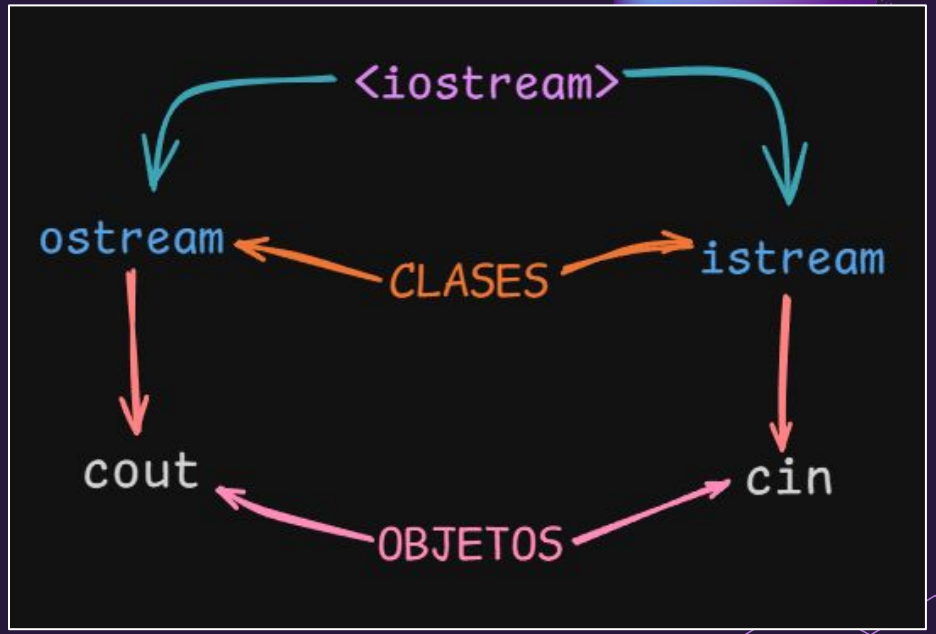
By: Candy

OBJETOS VS VARIABLES

Dicho de una manera medianamente técnica, un “Objeto” es la instancia de una clase.

Pero.. para que lo entiendan de manera práctica, pongamoslo así. Nosotros tenemos variables normales como “enteros” que son tipo “int” , como “decimales” que son tipo “double” o “float”, como “caracteres” que son tipo “char”; entonces, siguiendo esta analogía, un “Objeto” sería de tipo “Clase”.

Nosotros trabajamos previamente con los objetos “cout” (de la clase “ostream”) y “cin” (de la clase “istream”) para manejar la entrada y salida de los datos. No obstante, ahora utilizaremos otros para así trabajar directamente con los archivos en lugar de redireccionar.

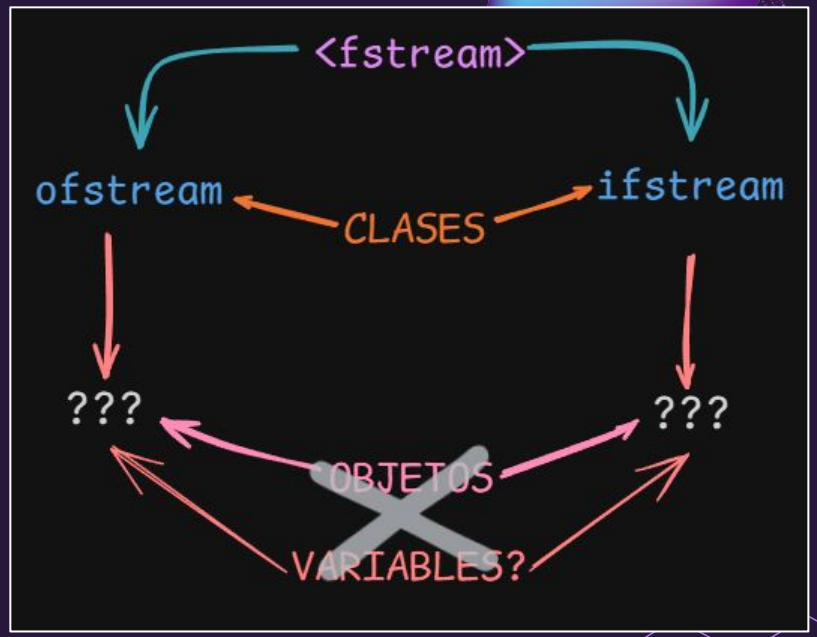


Candy

FSTREAM

De manera muy parecida con la libreria “iostream”, llega la librería “fstream”. Esta nos trae las clases “ofstream” (archivos de salida) e “ifstream” (archivos de entrada), para manejar salida y entrada de datos directamente con los archivos.
Pero entonces.. ¿por qué no aparecen sus objetos en la imagen?
Como ya habrán notado por la analogía anteriormente mencionada, nosotros ahora vamos a crear nuestros objetos! Pero tampoco se emocionen, pues será de una forma muy básica, tanto como para llamarlas variables.

Pues en lugar de hacer:
>> int numero;
// Para definir un entero
Deberíamos poder también hacer algo como:
>> ofstream arch;
// Para definir un archivo (en este caso de salida)



Nota:
> Existen más clases en esta librería las cuales no aplicaremos hasta tal vez mucho más a fondo en el curso, así que no se preocupen.

Candy

CONSTRUCTORES

A diferencia de una variable regular, una definición usual (sin asignación) para un objeto requiere de parámetros para su correcta inicialización, pues cada objeto se llega a componer de varios subdatos por dentro (dicho de una manera muy simple).

Para las clases “ofstream” y “ifstream”, ambos constructores tienen la siguiente forma:

```
(const char *nombre de archivo.ext, ios_base::openmode modo de apertura);
```

Archivo de Entrada →

```
ifstream archEntrada(“Archivo Fuente.txt”, ios::in);
```

Archivo de Salida →

```
ofstream archSalida(“Archivo Destino.txt”, ios::out);
```

Archivo de Salida (Añadidura) →

```
ofstream archXtra(“Archivo Destino.txt”, ios::app);
```

Candy

CONSTRUCTORES

Adicionalmente, en caso deseen inicializar los archivos mucho después, puede hacerlo implementando el método “open”.

```
.open(const char *nombre de archivo.ext, ios_base::openmode modo de apertura);
```

```
ifstream archEntrada;
ofstream archSalida,archXtra;
```

Archivo de Entrada → `archEntrada.open(“Archivo Fuente.txt”, ios::in);`

Archivo de Salida → `archSalida.open(“Archivo Destino.txt”, ios::out);`

Archivo de Salida (Añadidura) → `archXtra.open(“Archivo Destino.txt”, ios::app);`

Candy

VALIDACIÓN DE APERTURA

Al inicializar nuestras variables de archivo, puede que nos encontremos con que el archivo no esté en el directorio correcto, tenga otro nombre, etc. Por ello, el archivo no se podría abrir produciendo así errores que harían que nuestro programa se caiga. Para evitar ello, debemos SIEMPRE validar la apertura de cada uno de nuestros archivos. Para ello, luego de la inicialización utilizaremos el método ".is_open()".

```
ifstream archEntrada("Fuente.txt",ios::in);
if(not archEntrada.is_open()) {
    cout<<"ERROR: Falla de apertura del archivo 'Fuente.txt'."<<endl;
    exit(1);
}
```

```
ofstream archSalida("Destino.txt",ios::out);
if(not archSalida.is_open()) {
    cout<<"ERROR: Falla de apertura del archivo 'Destino.txt'."<<endl;
    exit(1);
}
```

Notas:

> Se valida si NO ESTÁ abierto para una mejor modulación.

> El mensaje del cout debe ser específico para distinguir cada archivo.

> "exit(1)" nos sale del programa indicando un estado de error.

Candy

BONUS: MODULACIÓN

Como notarán, si es que llegan a trabajar con varios archivos, la apertura se les hará extensa, por ende debería haber una forma de lograr resumirlo y generalizarlo para cada caso. Cabe resaltar que esta forma puede que no sea validada por el curso como tal, por lo que se recomienda preguntar a los evaluadores.

Entonces para esto, simplemente lo que haremos será una función que generalice el proceso, teniendo en cuenta que las funciones pueden devolver valores de todo tipo y esta analogía:
Si:
>> int numero = *valor entero*;
Entonces:
>> ifstream arch = *valor ifstream*;
>> ofstream arch = *valor ofstream*;

```
ifstream abrirArchivo_IFS(const char *nombArch){
    ifstream archIFS(nombArch,ios::in);
    if(archIFS.is_open()) return archIFS;
    cout<<"ERROR: Falla de apertura del archivo '"<<nombArch<<"'."<<endl;
    exit(1);
}
```

```
ifstream archEntrada = abrirArchivo_IFS("Fuente.txt");
```

Candy

DESTRUCTORES

Al igual que para una variable normal, los destructores se encargan de eliminar todo uso de la variable como tal. Estos usualmente se se inician automáticamente al finalizar un programa, bloque, función, etc. Pero a su vez, en algunos casos como este, se pueden llamar por métodos, por lo que refiriéndose a este contexto, el método se usaría para cerrar el archivo y ya nunca más referirlo a partir de esa variable.

ifstream

ofstream

```
arch.close();
```

Entonces esto nos lleva a hablar de los métodos!

Nota:
> Esto no se suele usar en el el curso, pero en caso de ser necesario simplemente se pondría al final de la función de donde se esté haciendo uso.

Candy

MÉTODOS .. COMPARTIDOS?!

Así como se ve, y pues es lógico viendo que las librerías son “iostream” y “fstream” , ¡algo en común debían tener!
Entonces..
>> TODOS LOS MÉTODOS (y sobrecargas) ESTUDIADOS PARA LOS OBJETOS DE LA CLASE “OSTREAM” (cout) SON LOS MISMOS QUE PARA LOS DE LA CLASE “OFSTREAM”!!

>> TODOS LOS MÉTODOS (y sobrecargas) ESTUDIADOS PARA LOS OBJETOS DE LA CLASE “ISTREAM” (cin) SON LOS MISMOS QUE PARA LOS DE LA CLASE “IFSTREAM”!!

```
archSalida.fill();
```

```
archSalida<<setw(10)<<”hola”<<endl;
```

```
archSalida.precision();
```

```
archEntrada.get();
```

```
archEntrada>>letra>>ws;
```

```
archEntrada.eof();
```

Candy

PARAMETRIZANDO

Dado que ustedes están EDITANDO directamente los archivos, si quisieran pasarlos como parámetros de una función deben hacerlo por REFERENCIA. Cabe resaltar que las variables de los archivos de lectura también son "editadas" pues el cursor de lectura es el que se va moviendo.

EJEMPLO:

```
void soyFuncional(ifstream &miEntrada,ofstream &miSalida){
    int numerito,funcionalidad = (int)'F'+(int)'U'+(int)'N'+
                                 (int)'C'+(int)'I'+(int)'O'+
                                 (int)'N'+(int)'A'+(int)'L';
    miEntrada>>numerito;
    miSalida<<(funcionalidad-numerito)<<endl;
}
```

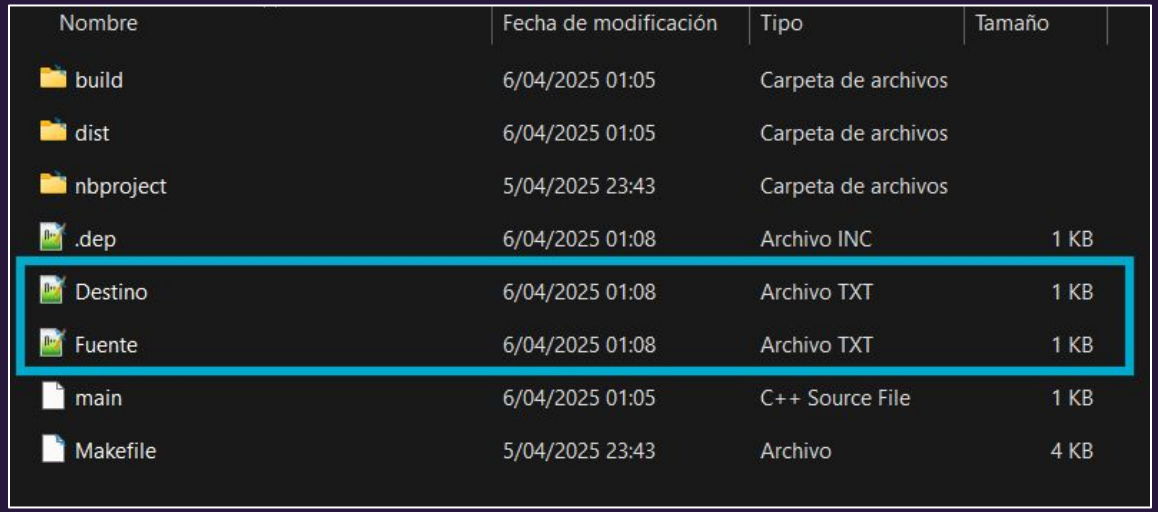
```
soyFuncional(archEntrada,archSalida);
```

Nota:
> Si se dan cuenta, y creo que ya era obvio, pueden variar el nombre de su variable de archivo dentro de la función pues son VARIABLES como tal!

Candy

DIFERENCIAS DE UBICACIÓN

Antes de realizar un ejemplo, debemos resaltar que la ubicación de los archivos será diferente, pues el programa toma y lee todo a partir de rutas relativas por lo que la "raíz" es el punto de inicio estándar para esto. Por ello, ahora los archivos no se ubicaran al costado del ejecutable, sino que serán puestos directamente en el directorio del proyecto.



Nombre	Fecha de modificación	Tipo	Tamaño
build	6/04/2025 01:05	Carpeta de archivos	
dist	6/04/2025 01:05	Carpeta de archivos	
nbproject	5/04/2025 23:43	Carpeta de archivos	
.dep	6/04/2025 01:08	Archivo INC	1 KB
Destino	6/04/2025 01:08	Archivo TXT	1 KB
Fuente	6/04/2025 01:08	Archivo TXT	1 KB
main	6/04/2025 01:05	C++ Source File	1 KB
Makefile	5/04/2025 23:43	Archivo	4 KB

Nota:
> Cabe resaltar que igualmente para los archivos de salida, no es necesario colocarlos la primera vez, pues se autogeneran.

Candy

EJEMPLO PRÁCTICO DE USO

```cpp
int main(int argc, char** argv) {
    // Apertura de Archivos
    ifstream archFuente("Fuente.txt",ios::in);
    if(not archFuente.is_open()) {
        cout<<"ERROR: Falla de apertura del archivo 'Fuente.txt'."<<endl;
        exit(1);
    }
    ofstream archDestino("Destino.txt",ios::out);
    if(not archDestino.is_open()) {
        cout<<"ERROR: Falla de apertura del archivo 'Destino.txt'."<<endl;
        exit(1);
    }
    // Declaracion de Variables
    int num;
    char digito;
    // Atoitizando digitos
    while(1){
        archFuente>>digito;
        if(archFuente.eof()) break;
        num = (int)digito - '0';
        archDestino<<"-> '"<<digito<<"' -> "<<num<<endl;
    }
    return 0;
}
```

TIPS

Como se puede ver en los comentarios, se recomienda que por cada bloque o función , modulen de esa forma su código; es decir:

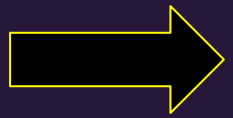
1) Apertura de Archivos
2) Declaración de Variables
..) Sus demás procesos..

Candy

EJEMPLO PRÁCTICO DE USO (Resultados)

Fuente.txt

```
1
2 3
4 5 6
7 8 9 0
```



Destino.txt

```
-> '1' -> 1
-> '2' -> 2
-> '3' -> 3
-> '4' -> 4
-> '5' -> 5
-> '6' -> 6
-> '7' -> 7
-> '8' -> 8
-> '9' -> 9
-> '0' -> 0
```

Candy

GRACIAS!

By: Candy