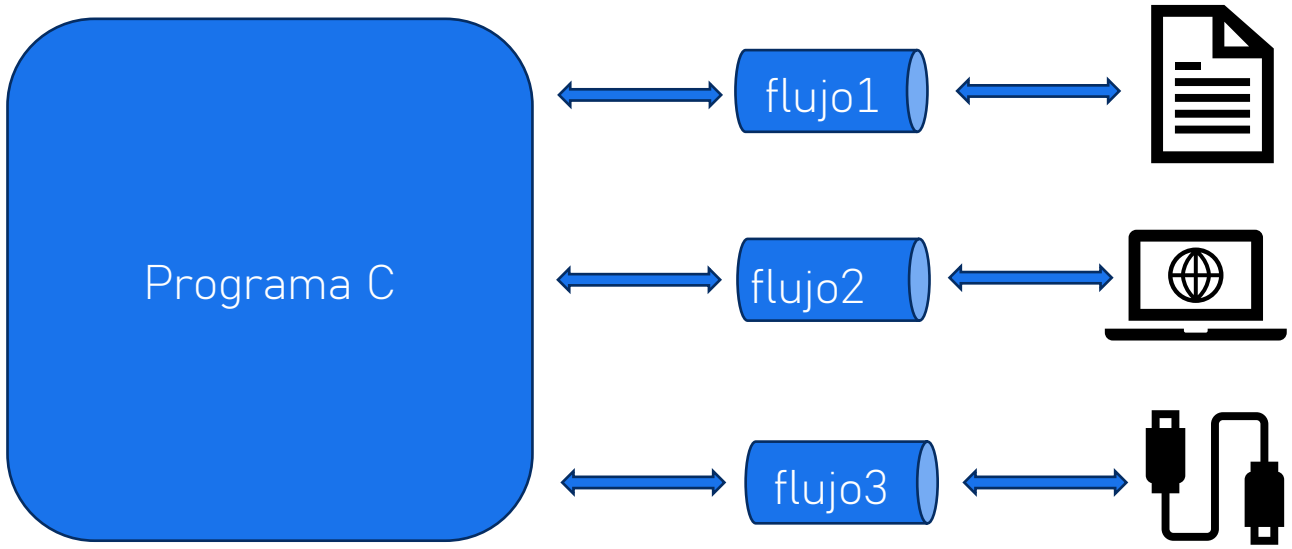
FLUJOS PRIMERA PARTE (STREAMS)

INFO II – PROF: WEBER

¿Qué es un *stream* en C?

En C, un **stream** (flujo) es una **abstracción** que representa una fuente o un destino de datos. Es como un "buffer" por donde entran o salen datos, sin que tengamos que preocuparnos por cómo se transmiten realmente.



- El motivo por el que existen los *streams* es que permiten al programador **leer o escribir datos sin preocuparse por el origen o destino real de esos datos**.
- Ya sea un archivo, la consola, un socket o un dispositivo, el código puede mantenerse igual, porque el acceso se hace a través de una interfaz común (*fopen*, *fscanf*, ***fprintf***, etc.).

Tipos de streams estándar en C

Nombre	Tipo	Significado común
stdin	Entrada	Teclado (lectura)
stdout	Salida	Pantalla (escritura normal)
stderr	Salida	Pantalla (pero para errores)

- Los streams estándar son variables globales que representan un puntero a una estructura FILE.
- O sea, stdin, stdout, stderr son del tipo FILE *.
- Estas variables apuntan a estructuras FILE internas que fueron creadas y abiertas automáticamente por el sistema cuando el programa arranca. No debemos hacer nada. Están ahí.

¿Y qué es FILE?

Es una estructura (struct) que representa un flujo de datos.
Esta estructura está oculta al usuario: no vamos a acceder a sus campos directamente. Trabajamos siempre a través de funciones como fgetc, fgets, **fprintf**, fread, etc.

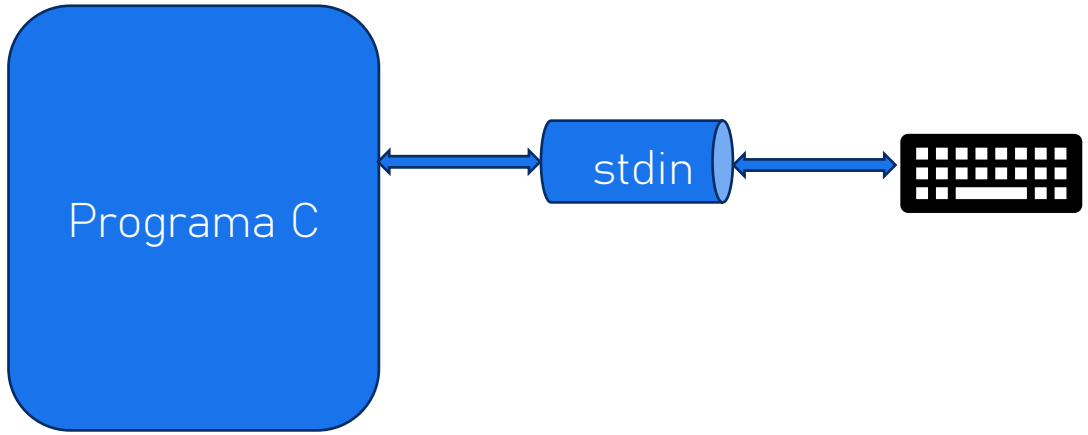
Con stdin ya hemos trabajado

```
Edad: 18
Tipo (A/B): Edad ingresada: 18
Tipo ingresado:

Press <RETURN> to close this window...
```

```c
#include <stdio.h>

int main() {
    int edad;
    char tipo;

    printf("Edad: ");
    scanf("%d", &edad);

    printf("Tipo (A/B): ");
    scanf("%c", &tipo);

    printf("Edad ingresada: %d\n", edad);
    printf("Tipo ingresado: %c\n", tipo);

    return 0;
}
```

¿Qué pasó?

- scanf("%d", &edad) lee el número 18 pero deja el '\n' en el buffer de stdin.
- scanf ("%c", &tipo) lo encuentra inmediatamente y lee eso como si fuera la entrada del nombre.



Con stdin ya hemos trabajado

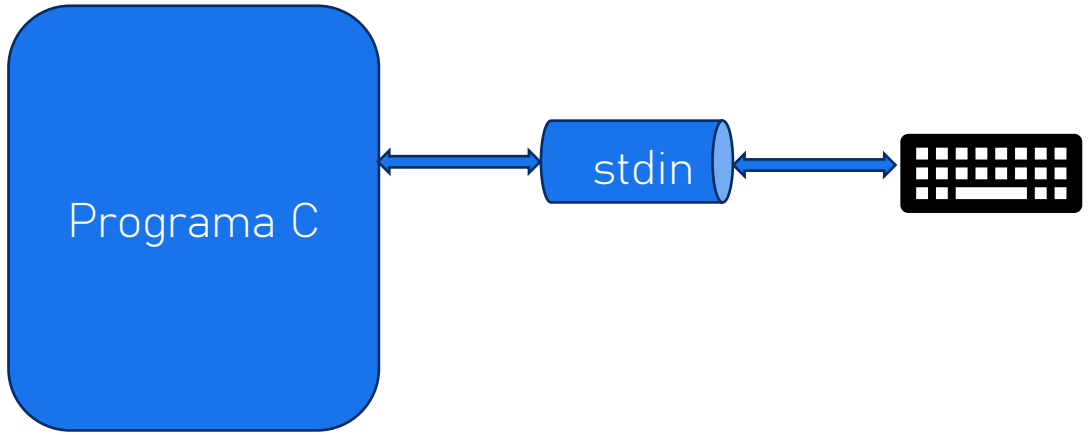
```c
#include <stdio.h>

int main() {
    int edad;
    char tipo;

    printf("Edad: ");
    scanf("%d", &edad);

    fflush (stdin);

    printf("Tipo (A/B): ");
    scanf("%c", &tipo);

    printf("Edad ingresada: %d\n", edad);
    printf("Tipo ingresado: %c\n", tipo);

    return 0;
}
```

```
Edad: 18
Tipo (A/B): A
Edad ingresada: 18
Tipo ingresado: A
Press <RETURN> to close this wi
```

```
Edad: 18
Tipo (A/B): Edad ingresada: 18
Tipo ingresado:

Press <RETURN> to close this window...
```

¿Cómo lo arreglabamos?

- Usando fflush (stdin)



Uso adecuado del fflush

- fflush es una función declarada en stdio.h
- Su prototipo completo es:
  - **int fflush (FILE *_stream_);**
- **¿Cuál es su función según el estándar?**
  - Vaciar (forzar) el **buffer de salida** asociado a un stream de salida.
  - Esto significa: si tenés datos en el buffer que aún no se han enviado al destino final (por ejemplo, a pantalla o a un archivo), **fflush** los envía inmediatamente.
- stdin …. ¿Es un flujo de salida o de entrada?
  - De entrada
  - ¿Entonces…?
    - En el estándar C, fflush(stdin) es comportamiento indefinido.
    - En sistemas como Windows con MSVC (Visual Studio), fflush(stdin) fue implementado como extensión para descartar el contenido pendiente en el buffer de entrada, es decir, vaciar lo que queda en stdin.
    - Por eso, en esos entornos, parece "arreglar" problemas como el salto de línea que queda tras un scanf.
    - Pero ese comportamiento no es portátil, no funciona igual en GCC, Linux o compiladores estándar, y no es la mejor practica.

Otros métodos de vaciar el flujo stdin

```c
#include <stdio.h>

int main() {
    int edad;
    char tipo,c;

    printf("Edad: ");
    scanf("%d", &edad);
    while ((c = getchar()) != '\n');
    printf("Tipo (A/B): ");
    scanf("%c", &tipo);
    while ((c = getchar()) != '\n');
    printf("Edad ingresada: %d\n", edad);
    printf("Tipo ingresado: %c\n", tipo);

    return 0;
}
```

Otro método alternativo es usar siempre la función **fgets**. Pero la veremos mas adelante.

Lo importante es recordar que la solución de fflush (stdin) NO es portable y NO es una buena practica.

Flujo redirigido desde stdout a un archivo mediante consola

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    if (argc < 2) {
        printf("Error: falta un número.\n");
        return 1;
    }

    char *fin;
    float valor = strtof(argv[1], &fin);

    if (*fin != '\0') {
        printf("Error: '%s' no es un numero valido.\n", argv[1]);
        return 1;
    }

    printf("[strtof] Valor convertido: %.2f\n", valor);
    return 0;
}
```

```
>argmain.exe 3.5
[strtof] Valor convertido: 3.50

>argmain.exe 3.5 > resultado.txt


>
```

Lo que hace el símbolo “>” es redirigir todo lo que va al stream stdout al archivo resultado.txt

Otra forma de ver el printf

- Prototipo completo de **printf**:
  - **int printf (const char *format , ...);**
- En este punto sabemos muy bien usar printf.
- Lo queremos remarcar es que printf interactúa directamente con el stream stdout.

- Prototipo completo de **fprintf**:
  - **int fprintf (FILE *stream , const char *format, ...);**
- Es igual a printf, pero debemos especificar con que flujo vamos a interactuar.
- Por lo tanto:.
  - fprintf (stdout, “Hola mundo”) ;
  - printf (“Hola mundo”);
  - Son idénticas.

Ir al QT. Ejemplo06. Mostrar cuando se va por stout y cuando por stderr

FIN