



Archivos



Archivos

- ▶ Los datos que cargamos hasta ahora se guardan en variables en la memoria principal (RAM) de la computadora, cuando se cierra el programa se pierden todos los datos
- ▶ Se pueden resguardar los datos en una memoria no volátil
 - ▶ Disco rígido
 - ▶ Pen-drive
 - ▶ Tarjeta SD
 - ▶ etc
- ▶ Esos datos se guardan físicamente en un archivo.
 - ▶ Ej archivo fuente .c
 - ▶ Archivos de Word
 - ▶ Archivo de imágenes
 - ▶ etc

Un archivo es un conjunto de datos almacenados en una memoria permanente.

Como se accede a los archivos

- ▶ Como tenemos que acceder a un dispositivo externo (hardware) disco, pendrive, etc. Cada uno podría tener una forma diferente de acceso para leer y escribir los archivos
- ▶ Como estamos en un lenguaje de alto nivel no tenemos que preocuparnos por esos detalles ya que existe una “abstracción” del lenguaje C que resuelve ese acceso mediante lo que se conoce como flujos o corrientes (stream).

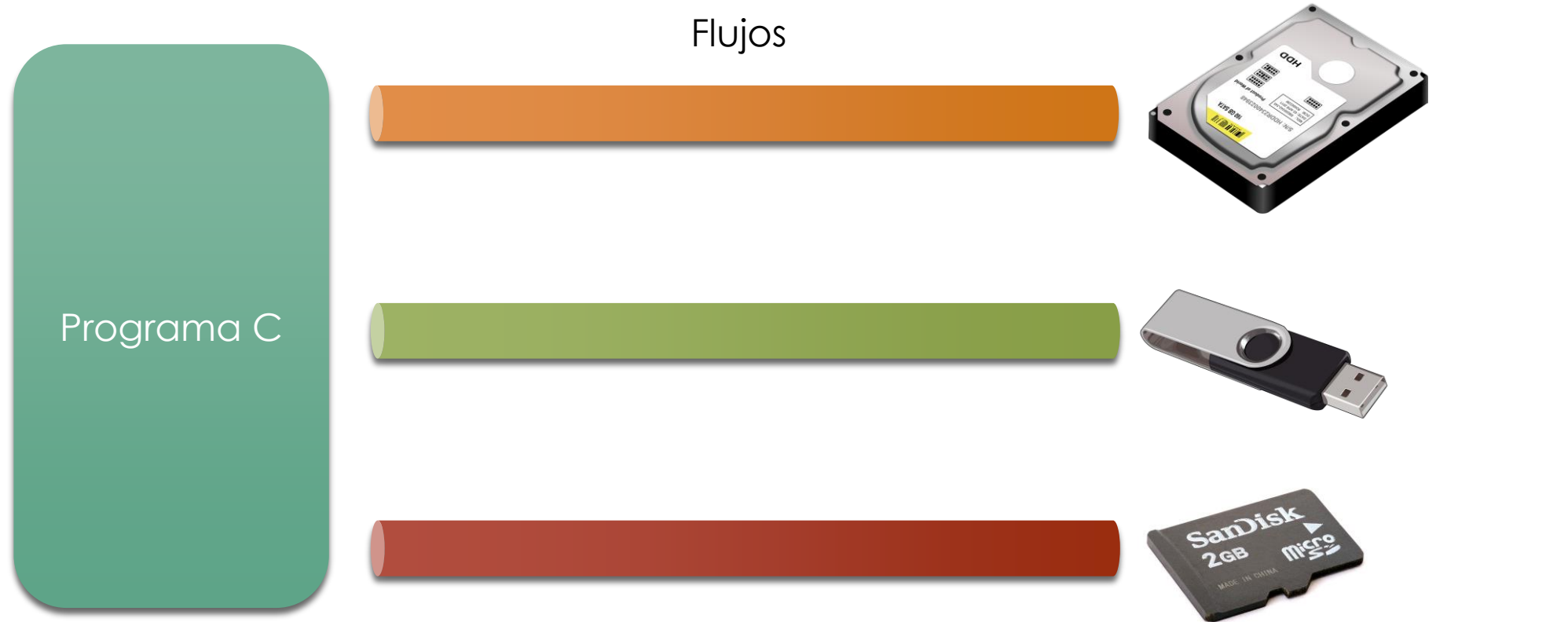
No importa donde grabo todos los dispositivos los puedo manejar de la misma forma.

Intermediario entre el programa y el hardware

El S.O. se encarga de manejar el hardware (driver)

Maneja el dispositivo en bajo nivel

Flujos en C



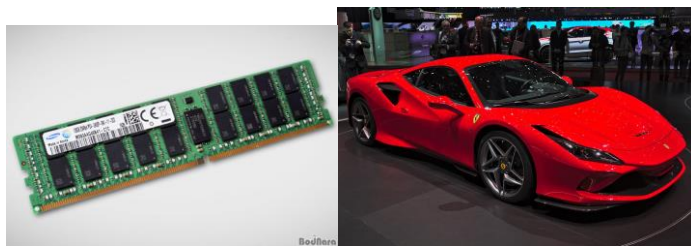
Flujos para el manejo de archivos en C

Dos tipos de flujos

- Flujos de Texto (solo caracteres hay un proceso de transformación para algunos caracteres por ejemplo el `\n` se guarda como `\r\n`)
 - Flujos Binario (se graban los bytes tal cual están en memoria)
-
- A cada flujo le doy un nombre diferente
 - Cada archivo va a tener su propio flujo
 - Si bien hay otras formas vamos a trabajar con archivos mediante el uso de un buffer intermedio

¿Porque se usa un buffer?

- ▶ Se usa porque tengo velocidades diferentes, el acceso a memoria es más rápida que el acceso al hardware
- ▶ En la memoria RAM la velocidad de acceso es del orden de los nano segundos 10^{-9} seg
- ▶ En un disco rígido mecánico la velocidad de acceso es del orden de los mili segundos 10^{-3} seg
- ▶ La diferencia está en el orden de 10^{-6}
- ▶ El acceso a la memoria RAM es un millón de veces más rápido que el acceso a un disco rígido



VS



- ▶ Con el buffer escribimos en memoria sin perder tiempo y cuando ese buffer se llena automáticamente se pasa al archivo de forma transparente para el programa.

Archivos con Buffer

Cuando escribimos o leemos un archivo no accedemos directamente al disco o donde este grabado, hay un intermediario que me permite desde el programa leer los datos y escribir directamente en la memoria principal y un proceso se encarga luego de llevarlo al archivo físico.

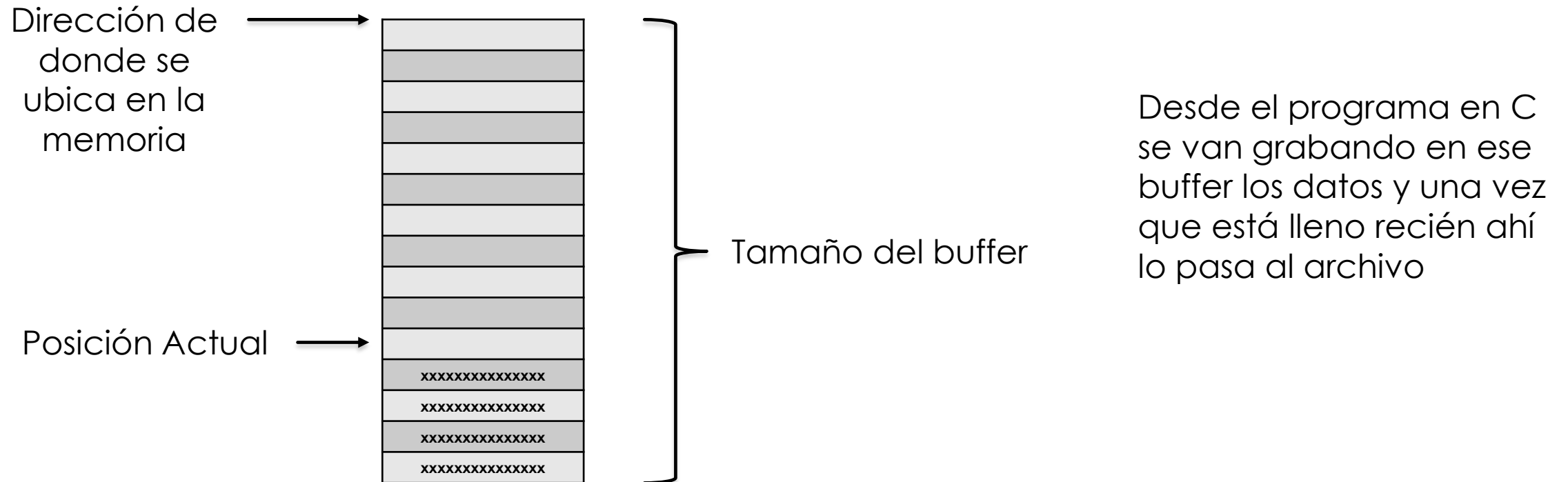
Esa área intermedia de memoria se denomina buffer

El buffer es un área de memoria que tiene cierta capacidad.

Esta memoria tiene un tipo de acceso FIFO (First In – First Out) es decir lo primero que grabo es lo primero que voy a leer, en castellano se conoce como cola.

Pensemos en la cola de un banco donde el primero que llega es el primero que atienden.

Para manejar un Buffer se necesitan varias cosas



Por cada archivo que tenga se debe gestionar toda la información del buffer y otras cosas mas...

Estructura para guardar los datos necesarios para administrar el flujo de los archivos

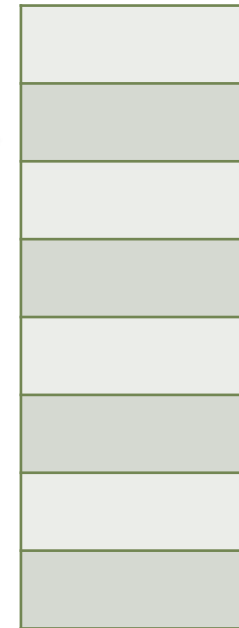
Definida en stdio.h

```
typedef struct _iobuf  
{  
    char*    _ptr;  
    int     _cnt;  
    char*    _base;  
    int     _flag;  
    int     _file;  
    int     _charbuf;  
    int     _bufsiz;  
    char*    _tmpfname;  
} FILE;
```

Estructura FILE

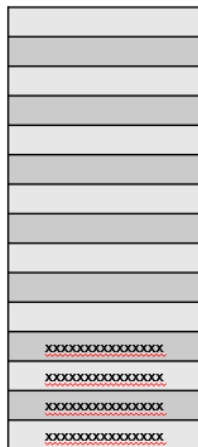
Tamaño Buffer
Posición Actual
Buffer
Flags
Descriptor del archivo
Indicador de posición en archivo
..
..

El sistema operativo maneja una tabla de Archivos Abiertos



Dirección de donde se ubica en la memoria

Posición Actual



Tamaño del buffer

Archivos Apertura

Las funciones para el manejo de archivos están en la biblioteca stdio.h

- ▶ Crear el Flujo
- ▶ Vincular al archivo físico
- ▶ Le asigno un nombre

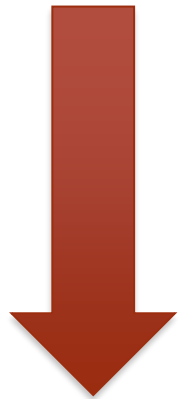
Que tipo de archivo quiero usar y que puedo hacer con ese archivo



```
FILE* fopen ("nombre del archivo", "modo de apertura");
```



Ruta y nombre del archivo físico
Es en el único lado que se usa el nombre del archivo
Sino pongo la ruta lo busca en la misma carpeta que el programa



Si la función se ejecuta correctamente devuelve la dirección de memoria de la variable tipo estructura FILE que maneja ese flujo

El * significa que es un puntero (guarda una dirección de memoria)
Si falla devuelve una dirección inválida (NULL) – Siempre luego del open chequeo si retornó NULL

Modos Apertura

En esta materia vamos a abrir a trabajar los archivos en forma secuencial para leer o escribir

- ▶ `r` `rt` `rb` se abre para lectura sino existe error
- ▶ `w` `wt` `wb` se crea para escritura, si existe lo destruye (ojo no avisa)
- ▶ `a` `at` `ab` modo de escritura donde se abre para agregar a continuación del último dato que está en archivo, sino existe lo crea

- ▶ `r+` `r+t` `r+b` se abre para lectura y escritura sino existe da error
- ▶ `w+` `w+t` `w+b` se crea para escritura y lectura, si existe lo destruye (ojo no avisa)
- ▶ `a+` `a+t` `a+b` modo de escritura y lectura donde se abre para agregar a continuación del último dato que está en archivo, sino existe lo crea

Estos modos requieren el uso de funciones para “desplazarse” por el archivo

- ▶ Libera la memoria , libera el file, libera la memoria dinámica del buffer, le avisa al sistema operativo que no usa más ese archivo, transfiere lo que queda en el buffer al archivo físico,

```
int fclose (FILE *);
```

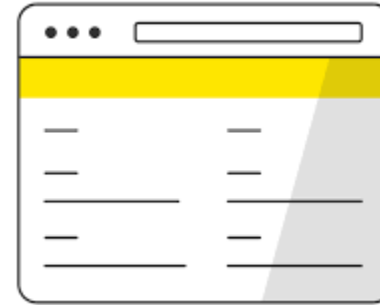
- ▶ Si devuelve un 0 lo cerró bien
- ▶ Puede devolver distinto de 0 si no lo pude cerrar, por ejemplo si saqué el pendrive donde tenía el archivo

- ▶ Sino conozco el formato con el que fue grabado no puedo leer la información porque no sé como interpretarla.
- ▶ En el archivo binario la información ocupa menos espacio que con el texto
- ▶ Lo grabo en el archivo con el mismo formato que lo tengo en la memoria, no necesito transformar la información.
- ▶ Grabo información de bloques fijos. Puedo acceder a una parte y recuperar información si todo ocupa lo mismo

Archivos Binarios

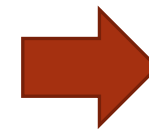


Archivo o Fichero



A cada ficha que guarda se la llama Registro

Cada registro tiene distintos campos con información



Tipo de dato definido con struct en C

Vamos a trabajar con Archivos binarios con registros de tamaño fijo

Todos los registros van a estar basados en la misma estructura

Si solo guardo un dato simple por ejemplo enteros, no necesito definir una estructura

Ventajas de usar Archivos Binarios

No necesito realizar transformación de datos, lo que grabo lo puedo pasar directo de la memoria y viceversa y usarlo directamente en mi programa

Si se trabaja con registros de ancho fijo por lo que puedo “moverme” entre los registros y acceder directamente hasta el que necesito.

Son más eficientes para almacenar números, por ejemplo si se quiere guardar este número fraccionario:
1735.45487

¿cuantos bytes ocuparían si fuera texto y cuantos bytes en binario?

10 bytes en texto porque son 10 caracteres

4 bytes en binario – variable float

Archivos Binarios - Escritura

Cuantos bytes voy a transferir,
partiendo de la dirección de inicio

```
fwrite (dir Dato, tam, registros, FILE *);
```

Dirección de inicio de donde
están los datos a escribir

Cantidad de registros a escribir

Partiendo de la dirección de inicio, lee X cantidad
de bytes y los envía al buffer

X se calcula como la multiplicación del 2do y 3er argumento

Archivos Binarios – Escritura ejemplo

Quiero crear un archivo con productos
Cada registro va a tener código, descripción y precio

1er Paso: Creo la estructura que va a definir los registros que se van a almacenar en el archivo

```
typedef struct
{
    int codigo;
    char descripcion[21];
    float precio;
}sProducto;
```

Archivos Binarios – Escritura ejemplo

Quiero crear un archivo con 3 productos
Cada registro va a tener código, descripción y precio

2do Paso: hago la apertura del archivo en modo escritura para que lo cree y genere el flujo

```
FILE * ap;  
ap = fopen("productos.dat", "wb");  
if (ap==NULL)  
{  
    printf("Error al abrir el archivo.");  
    getch();  
    exit(1);  
}
```

Archivos Binarios – Escritura ejemplo

Quiero crear un archivo con productos
Cada registro va a tener código, descripción y precio

3er Paso: declaro una variable del tipo de la estructura, e ingreso los datos por teclado

```
sProducto IngresarProducto();  
int main()  
{  
    sProducto prod;  
  
    .....  
    prod=IngresarProducto();  
}
```

Archivos Binarios – Escritura ejemplo

Quiero crear un archivo con productos
Cada registro va a tener código, descripción y precio

4to Paso: escribo, transfiero al archivo los datos que tengo en memoria

```
sProducto IngresarProducto();  
int main()  
{  
    sProducto prod;  
  
    .....  
  
    prod=IngresarProducto();  
    fwrite(&prod, sizeof(prod), 1, ap);  
  
    .....
```

```
typedef struct
```

```
{  
    int codigo;  
    char descripcion[21];  
    float precio;  
} sProducto;
```

```
sProducto IngresarProducto();
```

```
int main()
```

```
{  
    sProducto prod;  
    int i;  
    FILE * ap;  
    ap = fopen("productos.dat", "wb");  
    if (ap==NULL)  
    {  
        printf("Error al abrir el archivo.");  
        getch();  
        exit(1);  
    }  
    for (i=0; i<3; i++)  
    {  
        prod=IngresarProducto();  
        fwrite(&prod, sizeof(prod), 1, ap);  
    }  
    fclose(ap);  
    return 0;  
}
```

El Indicador de
posición en el
archivo apunta al
principio del
archivo



Crea productos.dat



```
typedef struct
```

```
{  
    int codigo;  
    char descripcion[21];  
    float precio;  
} sProducto;
```

```
sProducto IngresarProducto();
```

```
int main()
```

```
{  
    sProducto prod;  
    int i;  
    FILE * ap;  
    ap = fopen("productos.dat", "wb");  
    if (ap==NULL)  
    {  
        printf("Error al abrir el archivo.");  
        getch();  
        exit(1);  
    }  
    for (i=0; i<3; i++)  
    {  
        prod=IngresarProducto();  
        fwrite(&prod, sizeof(prod), 1, ap);  
    }  
    fclose(ap);  
    return 0;  
}
```

El Indicador de
posición en el
archivo apunta al
principio del
archivo



En memoria se completa los
datos del producto

prod

100	Perfume	890,75
prod.codigo	prod.descripcion	prod.precio

```
typedef struct
```

```
{  
    int codigo;  
    char descripcion[21];  
    float precio;  
}sProducto;
```

```
sProducto IngresarProducto();
```

```
int main()
```

```
{
```

```
    sProducto prod;
```

```
    int i;
```

```
    FILE * ap;
```

```
    ap = fopen("productos.dat", "wb");
```

```
    if (ap==NULL)
```

```
    {
```

```
        printf("Error al abrir el archivo.");
```

```
        getch();
```

```
        exit(1);
```

```
    }
```

```
    for (i=0;i<3;i++)
```

```
    {
```

```
        prod=IngresarProducto();
```

```
        fwrite(&prod, sizeof(prod), 1, ap);
```

```
    }
```

```
    fclose(ap);
```

```
    return 0;
```

```
}
```

El Indicador de posición en el archivo se mueve a la posición siguiente a los datos escritos

100 Perfume

890,75



Elementos de Programación

Escribe los datos en el archivo, partiendo de la dirección de prod a partir de donde indica el indicador de posición en el archivo

prod

100

Perfume

890,75

prod.codigo

prod.descripcion

prod.precio

```
typedef struct
```

```
{  
    int codigo;  
    char descripcion[21];  
    float precio;  
} sProducto;
```

```
sProducto IngresarProducto();
```

```
int main()
```

```
{
```

```
    sProducto prod;
```

```
    int i;
```

```
    FILE * ap;
```

```
    ap = fopen("productos.dat", "wb");
```

```
    if (ap==NULL)
```

```
    {
```

```
        printf("Error al abrir el archivo.");
```

```
        getch();
```

```
        exit(1);
```

```
    }
```

```
    for (i=0; i<3; i++)
```

```
    {
```

```
        prod=IngresarProducto();
```

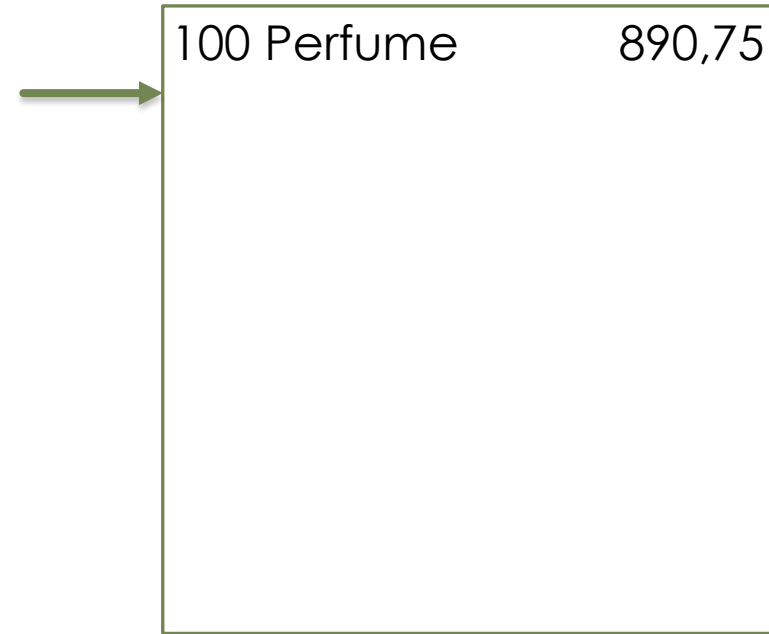
```
        fwrite(&prod, sizeof(prod), 1, ap);
```

```
    }
```

```
    fclose(ap);
```

```
    return 0;
```

```
}
```



En memoria se completan los datos de
otro producto

prod →	200	Algodon	120,50
	prod.codigo	prod.descripcion	prod.precio


```
typedef struct
```

```
{  
    int codigo;  
    char descripcion[21];  
    float precio;  
} sProducto;
```

```
sProducto IngresarProducto();
```

```
int main()
```

```
{
```

```
    sProducto prod;
```

```
    int i;
```

```
    FILE * ap;
```

```
    ap = fopen("productos.dat", "wb");
```

```
    if (ap==NULL)
```

```
    {
```

```
        printf("Error al abrir el archivo.");
```

```
        getch();
```

```
        exit(1);
```

```
    }
```

```
    for (i=0; i<3; i++)
```

```
    {
```

```
        prod=IngresarProducto();
```

```
        fwrite(&prod, sizeof(prod), 1, ap);
```

```
    }
```

```
    fclose(ap);
```

```
    return 0;
```

```
}
```

El Indicador de posición en el archivo se mueve a la posición siguiente a los datos escritos

100	Perfume	890,75
200	Algodon	120,50



Elementos de Programación



Escribe los datos en el archivo, partiendo de la dirección de prod a partir de donde indica el indicador de posición en el archivo

prod



200	Algodon	120,50
prod.codigo	prod.descripcion	prod.precio

```
typedef struct
```

```
{  
    int codigo;  
    char descripcion[21];  
    float precio;  
}sProducto;
```

```
sProducto IngresarProducto();
```

```
int main()
```

```
{  
    sProducto prod;  
    int i;  
    FILE * ap;  
    ap = fopen("productos.dat", "wb");  
    if (ap==NULL)  
    {  
        printf("Error al abrir el archivo.");  
        getch();  
        exit(1);  
    }  
    for (i=0;i<3;i++)  
    {  
        prod=IngresarProducto();  
        fwrite(&prod, sizeof(prod), 1, ap);  
    }  
    fclose(ap);  
    return 0;  
}
```

100 Perfume 890,75
200 Algodon 120,50



Elementos de
Programación

En memoria se completan los datos del
último producto

prod →	300	Peine	35,00
	prod.codigo	prod.descripcion	prod.precio

```
typedef struct
```

```
{  
    int codigo;  
    char descripcion[21];  
    float precio;  
} sProducto;
```

```
sProducto IngresarProducto();
```

```
int main()
```

```
{
```

```
    sProducto prod;
```

```
    int i;
```

```
    FILE * ap;
```

```
    ap = fopen("productos.dat", "wb");
```

```
    if (ap==NULL)
```

```
    {
```

```
        printf("Error al abrir el archivo.");
```

```
        getch();
```

```
        exit(1);
```

```
    }
```

```
    for (i=0; i<3; i++)
```

```
    {
```

```
        prod=IngresarProducto();
```

```
        fwrite(&prod, sizeof(prod), 1, ap);
```

```
    }
```

```
    fclose(ap);
```

```
    return 0;
```

```
}
```

El Indicador de posición en el archivo se mueve a la posición siguiente a los datos escritos

100	Perfume	890,75
200	Algodon	120,50
300	Peine	35,00



Elementos de Programación


Escribe los datos en el archivo, partiendo de la dirección de prod a partir de donde indica el indicador de posición en el archivo

prod

300	Peine	35,00
prod.codigo	prod.descripcion	prod.precio

```
typedef struct
```

```
{  
    int codigo;  
    char descripcion[21];  
    float precio;  
}sProducto;  
  
sProducto IngresarProducto();  
int main()  
{  
    sProducto prod;  
    int i;  
    FILE * ap;  
    ap = fopen("productos.dat", "wb");  
    if (ap==NULL)  
    {  
        printf("Error al abrir el archivo.");  
        getch();  
        exit(1);  
    }  
    for (i=0;i<3;i++)  
    {  
        prod=IngresarProducto();  
        fwrite(&prod, sizeof(prod), 1, ap);  
    }  
    fclose(ap);  
    return 0;  
}
```



100	Perfume	890,75
200	Algodon	120,50
300	Peine	35,00



Cierra el archivo liberando la memoria
del flujo y notificando al sistema
operativo

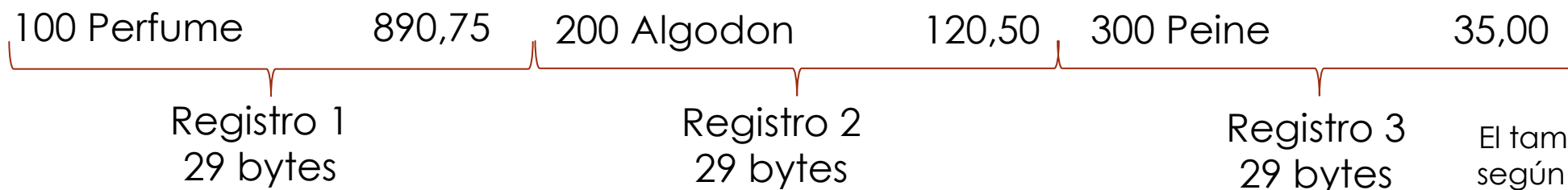
Cada registro ocupa lo mismo en el archivo sin importar los datos almacenados

Quedaron grabados 3 registros

100	Perfume	890,75
200	Algodon	120,50
300	Peine	35,00

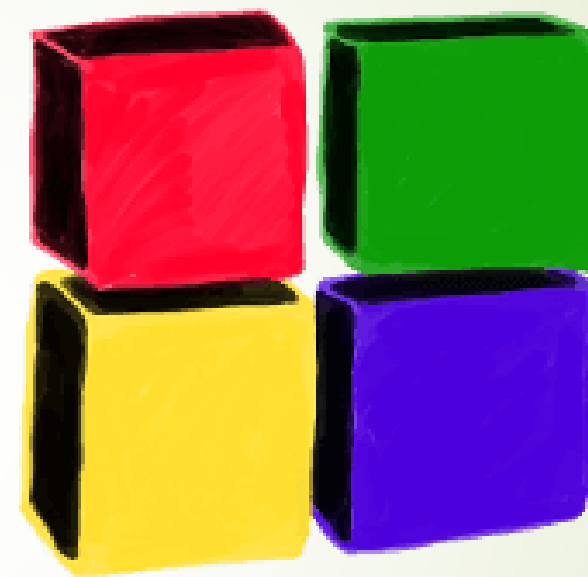
Se lo esquematiza como uno debajo del otro pero en realidad son bytes consecutivos grabados en el archivo

El tamaño de cada registro sale de sumar el peso de cada campo de la estructura



El tamaño en disco puede variar según como asigne el espacio el sistema operativo

EJEMPLO DE ESCRITURA



Code::Blocks

Cuantos bytes voy a leer

```
fread (dir Dato, tam, registros, FILE *);
```

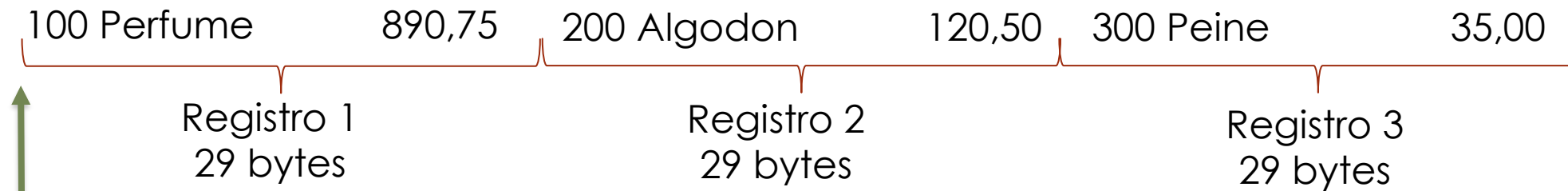
Dirección de inicio a partir de
donde voy a guardar los datos
en memoria

Cantidad de registros a leer

Partiendo del indicador de posición en el archivo
lee X cantidad de bytes y los copia en memoria a
partir de la dirección del dato

X se calcula como la multiplicación del 2do y 3er argumento

Archivos Binarios - Lectura



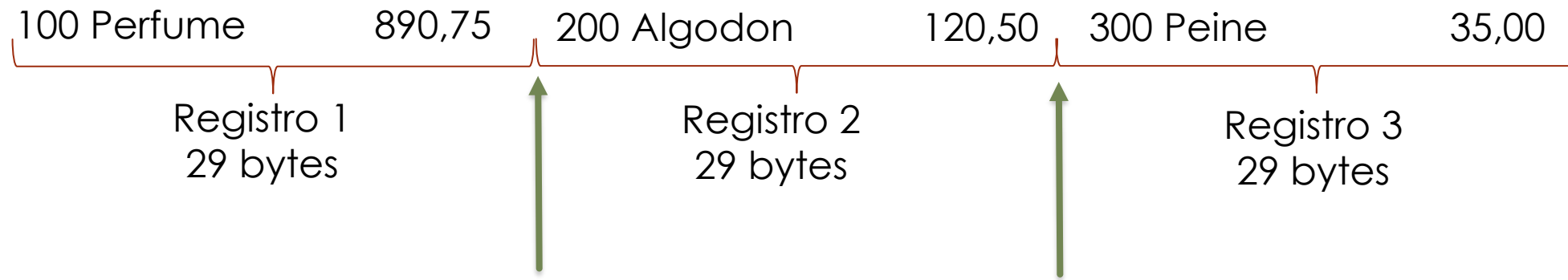
Apenas se abre el archivo para lectura el Indicador de posición en el archivo apunta al principio del archivo

Archivos Binarios - Lectura



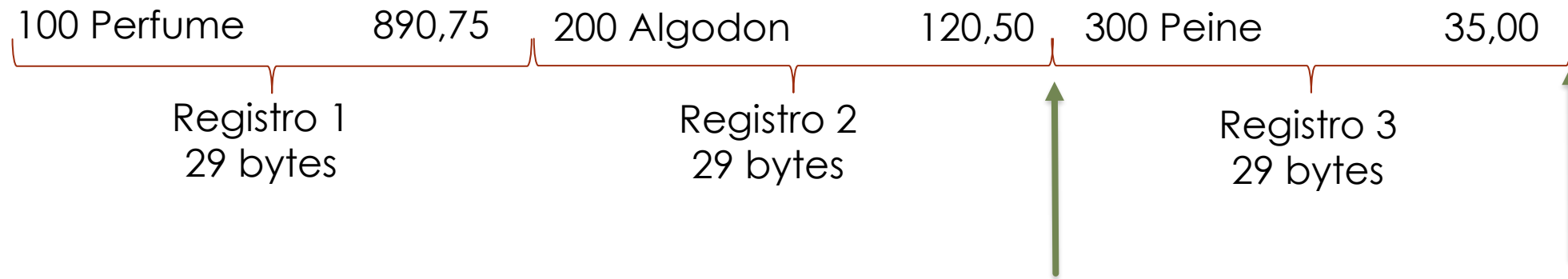
Con cada lectura
se va desplazando
a posición siguiente
de lo que leyó

Archivos Binarios - Lectura



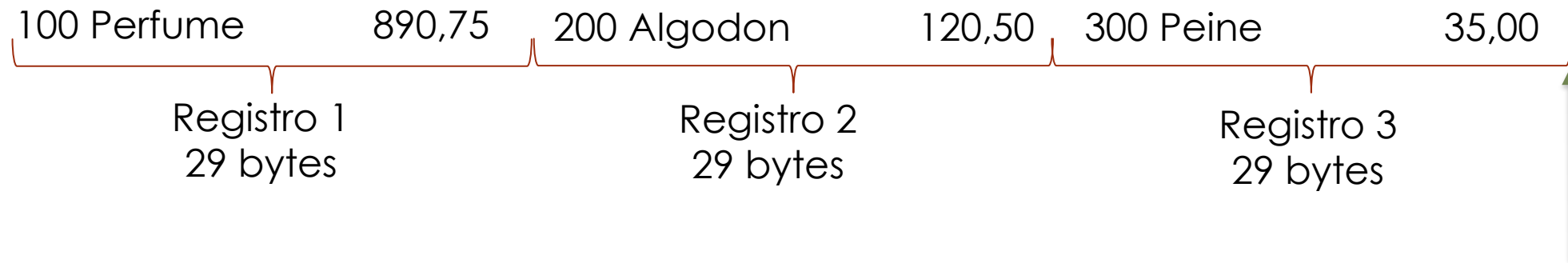
Con cada lectura
se va desplazando
a posición siguiente
de lo que leyó

Archivos Binarios - Lectura



Con cada lectura
se va desplazando
a posición siguiente
de lo que leyó

Archivos Binarios - Lectura



Cuando vuelve a leer como no hay mas registros, no guarda nada en memoria pero cambia un flag en la estructura FILE que indique se llegó al fin del archivo

Para saber si llegué al final del archivo LUEGO DE LEER puedo usar la función feof.

Archivos Binarios – Detección del fin de archivo

```
int feof (FILE *);
```



Devuelve 0 si NO llegó al final
del archivo

!=0 FIN DE ARCHIVO

CUIDADO!!!!
Solo lo puedo usar si antes
realice un fread sobre el mismo
archivo

Archivos Binarios – Ejemplo de Lectura

```
#include <stdio.h>
#include <stdlib.h>
typedef struct
{
    int codigo;
    char descripcion[21];
    float precio;
}sProducto;
```

```
int main()
```

```
{
```

```
    sProducto prod;
```

```
    int i;
```

```
    FILE * ap;
```

```
    ap = fopen("productos.dat", "rb");
```

Abro el archivo en modo lectura

```
    if (ap==NULL)
```

```
    {
```

```
        printf("Error al abrir el archivo.");
```

```
        getch();
```

```
        exit(1);
```

```
    }
```

Leo el primer registro

```
    printf("%8s %-20s %10s", "CODIGO", "DESCRIPCION", "PRECIO");
```

```
    fread(&prod, sizeof(prod), 1, ap);
```

```
    while (!feof(ap))
```

a leer

Mientras aún tenga datos el archivo los muestro y vuelvo a leer

```
    {
```

```
        printf("\n%8d %-20s %10.2f", prod.codigo, prod.descripcion, prod.precio);
```

```
        fread(&prod, sizeof(prod), 1, ap);
```

```
    }
```

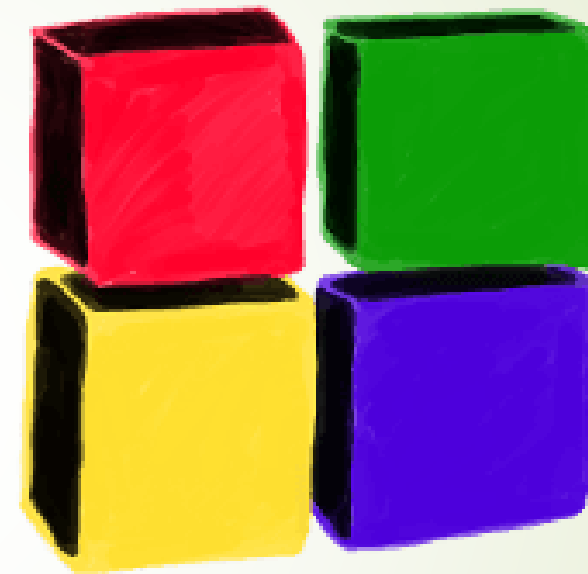
```
    fclose(ap);
```

Cierro el archivo

```
    return 0;
```

```
}
```

EJEMPLO DE LECTURA



Code::Blocks

¿Como enviar un listado a una persona que no tiene el programa?

Para leer un archivo binario necesito sí o sí tener un programa que conozca como fueron grabados esos datos

Si quiero mandar por ejemplo la lista de mis productos con mis precios a un cliente debería pasar también el programa lector



SOLUCIÓN: Usando un archivo de texto puedo exportar los datos para generar un listado visible en planillas de cálculo

Exportar datos

Para escribir en un archivo de texto de forma fácil vamos a usar la siguiente función:

```
fprintf (FILE *, "texto con formato", lista de variables);
```



Flujo donde quiero enviar los datos



Igual al printf
sino pongo formatos en texto
no pongo variables

Exportar datos

```
int main()  
{
```

```
    sProducto prod;
```

```
    int i;
```

```
    FILE *ap, *exportar;
```

```
    ap = fopen("productos.dat", "rb");
```

```
    exportar = fopen("productos.csv", "wt");
```

```
    if (ap==NULL || exportar==NULL)
```

```
    {
```

```
        printf("Error al abrir el archivo.");
```

```
        getch();
```

```
        exit(1);
```

```
    }
```

```
    fprintf(exportar, "CODIGO;DESCRIPCION;PRECIO\n");
```

```
    fread(&prod, sizeof(prod), 1, ap);
```

```
    while (!feof(ap))
```

```
    {
```

```
        fprintf(exportar, "%d;%s;%.2f\n", prod.codigo, prod.descripcion, prod.precio);
```

```
        fread(&prod, sizeof(prod), 1, ap);
```

```
    }
```

```
    fclose(ap);
```

```
    fclose(exportar);
```

```
    return 0;
```

Vamos a generar un archivo de texto con extensión csv (archivo con campos separados por un delimitador)

Debo chequear si se abrieron todos los archivos

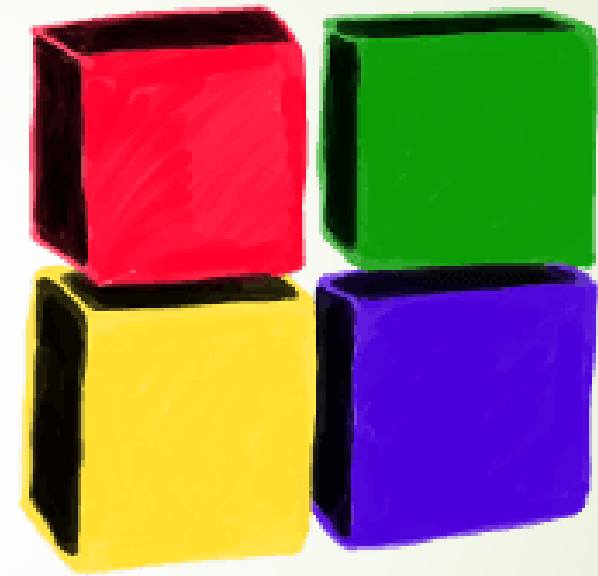
Escribo una línea de títulos

Escribo una línea por cada registro del archivo binario

Cierro los dos archivos



EJEMPLO PARA EXPORTAR



Code::Blocks

Flujos en C

3 flujos son creados por defecto

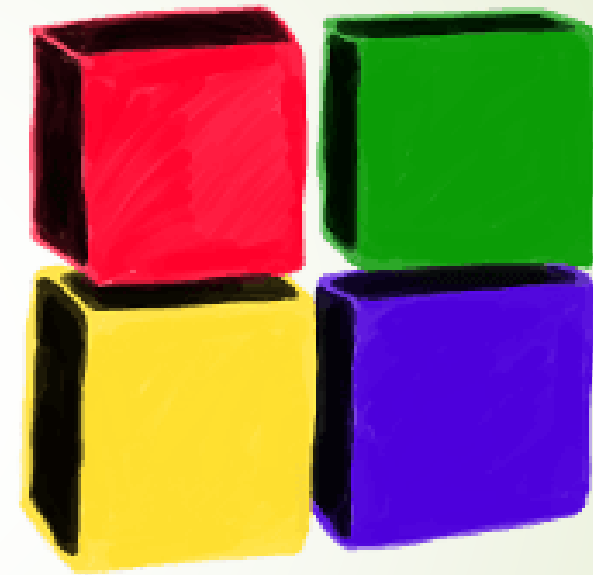
`stdin` flujo de entrada (teclado)

`stdout` flujo de salida (pantalla) , `printf`, `puts`, `putchar`

`stderr` flujo de salida (pantalla) para errores

Si al programa anterior le cambio en el `fprintf` y en lugar del puntero al archivo pongo `stdout` lo muestra por pantalla!!

EJEMPLO PARA
ENVIAR AL FLUJO
DE PANTALLA



Code::Blocks

Archivos – Resumen

FILE *	Puntero a la estructura de control del flujo para comunicarnos con el archivo
fopen	Crea el flujo y retorna el puntero
fclose	Cierra el flujo y libera la memoria
fread	Lectura de Archivos Binarios
fwrite	Escritura de Archivos Binarios
feof	Detección del fin de archivo
fprintf	Escritura con formato en archivos de texto



Elementos de
Programación

¡Muchas gracias!



DIIT
Departamento de Ingeniería e
Investigaciones Tecnológicas
Universidad Nacional de La Matanza