Parámetros de entrada

... por línea de comandos

Recordemos: ejecutar un programa

1. El S.O. carga el programa compilado **en memoria**

2. El S.O. llama a la (única) función **`main`** existente

3. El programa ejecuta, y, al terminar su **main**, devuelve un **int** con el resultado de su ejecución

Interacción usuario - programa

- **¿Cómo le paso información al programa?**
- **Mientras ejecuta:**
  - **cin** para leer datos escritos por el usuario en la terminal
  - **cout** para escribir salida de texto en la terminal
  - ...ya los usamos extensamente
- **Antes de ejecutarlo:**
  - **parámetros de línea de comandos**
    - **¿?**
  - variables de entorno (*no las vamos a ver...*)

Línea de comandos: ejemplo cp

- El comando **cp fuente destino** duplica el archivo **fuente** con el nuevo nombre **destino**

- ¿Cómo hace **cp** para recibir esos dos parámetros?
  - ¿Se puede usar **cin** / **cout**? ¿Por que?

Línea de comandos: ejemplo cp

- El comando **cp fuente destino** duplica el archivo “fuente” con el nombre “destino”

- ¿Cómo hace **cp** para recibir esos dos parámetros?
  - ¿Puedo usar **cin** / **cout**? ¿Por que?
  - Hasta que yo no apreté “enter” luego de tipear
    cp fuente destino
    el programa no empezó a ejecutar

Línea de comandos: ejemplo cp

- El comando **cp fuente destino** duplica el archivo “fuente” con el nombre “destino”

- ¿Cómo hace **cp** para recibir esos dos parámetros?
  - ¿Puedo usar **cin** / **cout**? ¿Por que?
  - Hasta que yo no apreté “enter” luego de tipear **cp fuente destino**, el programa no empezó a ejecutar

- Y entonces, **¿qué hago?**

El main SI recibe parámetros

int main(**int** cantidad, **char**** parametros);

- **int cantidad**: número de parámetros recibidos por consola
- **char** parametros**: arreglo (*) de arreglos (*) de **char**
  - **un arreglo de char es la versión primitiva de un string**
    - *El modificador * indica que el tipo es un puntero. Para nosotros, es lo mismo que un array.*
  - **parametros[0]** es siempre el string con el nombre del programa
  - **parametros[1] ... parametros[cantidad-1]** son los parámetros de línea de comandos.

- Por convención C, se suele llamar
  - **cantidad** como **argc**
  - **parametros** como **argv**

- **¿Por qué hasta ahora nos dejaba omitir los parámetros?**

Ejemplo: cp main.cpp CMakeLists.txt tp1/

- **cantidad** = 4
- **parametros**[0] = “cp”
- **parametros**[1] = “main.cpp”
- **parametros**[2] = “CMakeLists.txt”
- **parametros**[3] = “tp1/”

Ejercicio 1

Escribir un programa que imprima por pantalla todos los parámetros recibidos.

Ejemplo: ./programa A B C ... Z

```
Usted ingresó 27 parámetros.

El parámetro 0 es: “programa”.
El parámetro 1 es: “A”.
...
El parámetro 26 es: “Z”.

Hasta luego.
```

Convirtiendo parámetros a string / int / float

- El arreglo “parámetros” sólo almacena texto
  - Y en un formato **muy** primitivo: **char***

- ¿Cómo convierto parametros[i] a distintos tipos de variable?
  - **int** n = **atoi**(parametros[i]);
  - **float** f = **atof(**parametros[i]);
    double f = **atof(**parametros[i]);
  - **string: simplemente asignar**
    string s = parametros[i];
    - C++ sabe cómo convertir un **char*** en **string**
- **Usar** #**include** <**cstdlib**>

Ejercicio 2

Escriba un programa que calcule sumatorias y productorias de los números pasados como parámetro por línea de comandos.

Ejemplos:

```
./programa sumatoria 1 2 3 4 5.5
La sumatoria es: 15.5

./programa productoria 1 2 3 4 5.5
La productoria es: 132
```

*Pueden asumir que los números ingresados serán todos **float/double**.*
*La cantidad de números es variable (podría ser **cero**).*
***Ejecutarlo desde terminal** en la carpeta del proyecto (para pruebas).*