# Plantilla Informe Técnico Taller de Programación

Diego Caro[1], Daniela Opitz[1]

[1]Universidad del Desarrollo

***Resumen*** **- El resumen es una descripción completa y concisa del trabajo que se realizó y se presenta en el documento. Debe de mostrar los puntos claves del tema que se trata, y no ser mayor de 300 palabras (o bien del límite que se imponga). Generalmente, se presenta la motivación del trabajo, por qué nos importa el tema a tratar y los resultados (si la problemática o el tema que tratamos es ampliamente conocido podemos obviar esta parte; sin embargo, si el problema no es obviamente interesante es mejor colocar la motivación antes de proseguir). Recordemos que sabemos lo qué sabe o no nuestro lector**.

Debemos incluir el problema que tratamos de resolver y explicar nuestra solución de manera simple. Debemos mencionar qué hicimos para resolver o atacar el problema (usamos simulaciones, construcción de prototipos, modelos y de qué clase, análisis de datos, etc.),

Note que el resumen no es una copia del enunciado, sino de las ideas principales que mostramos en detalle en el documento.

## I. INTRODUCCIÓN

El objetivo principal de la introducción es contextualizar el texto presentado. En esta sección se describe el alcance del documento, y se da una explicación más detallada de los temas presentados en el resumen. Adicionalmente, puede presentar antecedentes del trabajo que realizamos que son importantes para el desarrollo del tema mismo.

En la introducción deberemos tratar y ampliar los puntos claves que presentamos en el resumen nuevamente. Sin embargo, debemos cuidar de no redundar, y de no presentar la información de manera superficial.

Siempre se debe de presentar el problema o el objeto a desarrollar, por ejemplo, un producto, una especificación, un algoritmo, un método, una investigación científica, o una revisión bibliográfica.

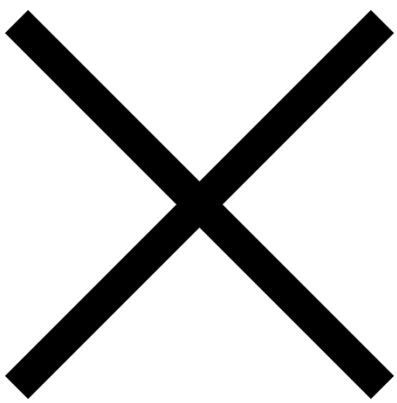

**Figura 1**: Ejemplo de figura en una

## II. DESARROLLO DE LA SOLUCION

Para el curso de Taller de Programación, en esta sección debe describir de manera simple y clara los desafíos más importantes que debió resolver para desarrollar su solución.

Cada debe ser explicado en detalle pero sin exagerar. Sí utilizó métodos no vistos en clases es en esta sección donde debe especificar su origen y funcionamiento básico.

Utilice la estructura que más le acomode para que su explicación sea sencilla de leer. Utilice secciones y subsecciones numeradas y refiérase a ellas cuando las necesite.

- Desafío 1
- Desafío 2

….

Puede incluir figuras, tablas y secciones de código. Asegúrese de numerarlas y describirlas correctamente.

- **Figuras**

Puede generar figuras que ocupen una columna (*ver Figura 1.*). Debe debe indicar una descripción de la figura) e incluir una cita de la figura en el texto.

- **Tablas**

Puede usar tablas con el menú insertar. Recuerde que al igual que las figuras debe incluir una descripción y citar la tabla en el texto. Sin embargo, debe incluir la descripción de las tablas sobre la tabla (ver Tabla 1).

**Tabla 1:** Descripción de una tabla

| Table Head | Table Column Head | | |
|---|---|---|---|
| | ***Table column subhead*** | ***Subhead*** | ***Subhead*** |
| copy | More table copy[a] | | |

Recuerde que si utiliza material externo para hacer su informe, debe citarlo. Para citar debe incluir la referencia (nombre de autores, título del trabajo, etc…). Además, debe citar las referencias en el texto, por ejemplo: **`Eason et al. [1] afirma que: ..`**

- **Incluyendo código**

Para incluir código se recomienda utilizar una tabla de una celda, y dentro de esta celda incluir el código. Para que el código sea visible, puede utilizar el sitio web http://hilite.me/ y pegar el texto formateado (en colores y negrita). Recuerde que también debe citar el código (por ejemplo, ver Figura 2).

```
def fib(n):
    if n < 2: return n
    return fib(n - 1) + fib(n - 2)
```

**Figura 2:** Versión recursiva para calcular Fibonacci

## III VALIDACION DE LA SOLUCION

En esta sección debes indicar por qué tu código resuelve el problema. Tus validaciones deben ser rigurosas y convincentes, considerando no sólo casos triviales, si no también casos límites. Lo más importante es demostrar que tu código resuelve correctamente el problema, sin importar como se escoja la entrada.

## IV ERRORES MAS COMUNES

1. Indicar la idea principal de manera poco clara. Cuando se omite la idea principal, el lector debe inferir desde tu código cuál es la intención detrás, lo que les hace perder tiempo y volverse más gruñones (y además, se vuelve más quisquilloso con los errores o typos en el documento). Si incluyes la idea principal, estás haciendo más fácil la lectura y haciendo que los lectores digan “Wow, el/ella entendió el concepto”, y luego es más probable obtener todo el puntaje en la pregunta.

2. Hacer una validación poco informativa. Indicar de manera informal o imprecisa la validación de tu solución podría ser indicio de que el algoritmo no resuelve el problema, o de que no se sabe por qué funciona, o bien, funciona para sólo unos casos y no todos.

3. Añadir dos hojas llenas de código en Python, en vez de escribir solo la sección que interesa para entender el algoritmo. Recuerda que el código final debes entregarlo en un archivo aparte o subirlo a githuh e indicar el link.

## III. Referencias

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.