

# Trabajos de Algoritmos y Esctructura de Datos

Esp. Ing. José María Sola, profesor.

Revision 0.1.1

Abr 2017

---

---

---

# Table of Contents

1. Introducción .....	1
2. Requisitos Generales para las Entregas de las Resoluciones .....	3
2.1. Requisitos de Tiempo .....	3
2.2. Requisitos de Forma .....	3
2.2.1. Repositorio del Equipo en GitHub .....	3
2.2.2. Front Page del Equipo .....	4
2.2.3. Carpetas para cada Resolución .....	4
2.2.4. Header Comments (Comentarios Encabezado) .....	5
2.2.5. Front Page de la Resolución .....	5
2.2.6. Problemas y Soluciones .....	6
3. Trabajo #0 — "Hello, World!" en C++ .....	7
3.1. Objetivos .....	7
3.2. Entrega .....	7
3.3. Secuencia de Tareas .....	7
4. Trabajo #1 — Adición .....	9
4.1. Problema .....	9
4.2. Entregables .....	9
5. Trabajo #2 — Mayor de dos Números .....	11
5.1. Problema .....	11
5.2. Entregables .....	11



## Introducción

---

Este documento presenta los enunciados de los trabajos prefijados del curso. Su resolución es grupal, salvo que se indique la contrario.



## Requisitos Generales para las Entregas de las Resoluciones

---

Cada trabajo tiene sus requisitos particulares de entrega de resoluciones, esta sección indica los requisitos generales, mientras que, cada trabajo define sus requisitos particulares.

Una resolución se considera **entregada** cuando cumple con los **requisitos de tiempo y forma** generales, acá descriptos, sumados a los particulares descriptos en el enunciado de cada trabajo.

### 2.1. Requisitos de Tiempo

Cada trabajo establece la fecha y hora límite de entrega, los commits realizados luego de ese instante no son tomados en cuenta para la evaluación de la resolución del trabajo.

### 2.2. Requisitos de Forma

Requisitos de forma del repositorio, las carpetas de las resoluciones, y los encabezados de los archivos fuente.

#### 2.2.1. Repositorio del Equipo en GitHub

La entrega de cada resolución debe realizarse a través de *GitHub*, para ello, a cada equipo se le asigna un **repositorio privado**. Lugo, el equipo debe crear una carpeta particular para cada resolución.

### **2.2.2. Front Page del Equipo**

En la raíz del repositorio cada equipo debe diseñar un archivo `readme.md` que actúe como front page del equipo. Debe estar escrito en notación *Markdown* y debe tener, como mínimo, la siguiente información:

- Signatura.
- Curso.
- Año de cursada, y cuatrimestre si corresponde.
- Número de equipo.
- Nombre del equipo (opcional).
- Integrantes del equipo actualizados, ya que, durante la transcurso de la cursada el equipo puede cambiar:
  - Usuario *GitHub*.
  - Legajo.
  - Apellido.
  - Nombre.

### **2.2.3. Carpetas para cada Resolución**

La resolución de cada trabajo debe tener su propia carpeta en el repositorio del equipo.

Además de los productos solicitados por cada trabajo, la carpeta **sí debe incluir**:

- un archivo `readme.md` que actúe como front page de la resolución.

Para facilitar el desarrollo se **recomienda incluir**:

- un archivo `.gitignore`.
- un archivo `makefile`.
- archivos tests.

Por último, la carpeta **no debe incluir**:

- archivos ejecutables.



- archivos intermedios producto del proceso de compilación o similar.

El siguiente patrón como la carpeta se **debe nombrar**:

ÚltimosTresDígitosDelCurso-DosDelEquipo-DosDígitosNúmeroTrabajo-  
NombreTrabajo

**Example 2.1. Nombre de carpeta**

051-02-00-Hello

## **2.2.4. Header Comments (Comentarios Encabezado)**

Todo archivo fuente de debe comenzar con un comentario que indique el “Qué”, “Quiénes”, “Cuándo” :

```
/* Qué: Nombre
 * Breve descripción
 * Quiénes: Autores
 * Cuando: Fecha de última modificación
 */
```

**Example 2.2. Header comments**

```
/* Stack.h
 * Interface for a stack of ints
 * JMS
 * 20150920
 */
```

## **2.2.5. Front Page de la Resolución**

Cada de resolución debe contar con un archivo `readme.md`, escrito en *Markdown* que contenga, como mínimo, la siguiente información:

- Número de equipo.

- Nombre del equipo (opcional).
- Autores:
  - Usuario github.
  - Legajo.
  - Apellido.
  - Nombre.
- Número y título del TP.
- Transcripción del enunciado.
- Hipótesis de trabajo que surgen luego de leer el enunciado.

### **2.2.6. Problemas y Soluciones**

Todos los archivos `readme.md` que actúan como *Front Page* de la resolución, deben contener una descripción del *análisis del problema* y una *síntesis de la solución*.

Luego del enunciado y las hipótesis, cada `readme.md` debe contener:

- Etapa de Análisis:
  - Entradas: nombres y tipos de datos.
  - Proceso: nombre descriptivo.
  - Salidas: nombres y tipos de datos.
- Etapa de Síntesis:
  - Léxico del Algoritmo.
  - Representación visual o textual del Algoritmo.

---

# 3

## Trabajo #0 — "Hello, World!" en C++

---

Este es un trabajo individual.

### 3.1. Objetivos

- Demostrar con, un programa simple, que se está en capacidad de editar, compilar, y ejecutar un programa C++.
- Contar con las herramientas necesarias para la resolución de los trabajos.

### 3.2. Entrega

- A las 13:00 del día de la segunda clase.

### 3.3. Secuencia de Tareas

1. Solicitar inscripción al Grupo Yahoo.
2. Registrarse en GitHub
3. Crear un repositorio público con el nombre `cppHelloWorld`.
4. Seleccionar, instalar, y configurar un compilador C++14.
5. Probar el compilador con un programa `hello.cpp` que envíe a `cout` la línea `hello, world!` o similar.
6. Ejecutar el programa, y capturar su salida en un archivo de texto `output.txt`.
7. Publicar `hello.cpp` y `output.txt` en GitHub.
8. Escribir y publicar el archivo `readme.md` del proyecto.

9. Enviar a [UTNFRBAAED@yahoogleups.com](mailto:UTNFRBAAED@yahoogleups.com)<sup>1</sup> usuario y repositorio GitHub.

---

<sup>1</sup> <mailto:UTNFRBAAED@yahoogleups.com>

---

# 4

## Trabajo #1 — Adición

---

### 4.1. Problema

Escribir los pasos para mostrar la suma de dos números que ingresa el usuario.

### 4.2. Entregables

- Carpeta: Adición
- `readme.md`
- Implementación: `Adición.cpp`.



---

# 5

## Trabajo #2 — Mayor de dos Números

---

### 5.1. Problema

Dado dos números informar cuál es el mayor.

### 5.2. Entregables

- Carpeta Mayor
- `readme.md`
- Implementación: `Mayor.cpp`.

