Trabajos de Algoritmos y Esctructura de Datos

Esp. Ing. José María Sola, profesor.

Revision 1.0.0
May 2017.

Table of Contents

1

Introducción

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

2

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 repostiorio, 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:

- Asignatura.
- 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 sigueinte 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**. Consta del diagrama del *Modelo IPO* con:
  - Entradas: nombres y tipos de datos.
  - Proceso: nombre descriptivo.
  - Salidas: nombres y tipos de datos.
- **Etapa de Diseño**. Consta del algoritmo que define el método por el cual el proceso obtiene las salidas a partir de las entradas:
  - Léxico del Algoritmo.
  - Representación visual o textual del Algoritmo.

La resolución incluye archivos fuente que forman el programa que implementan el algoritmo definido. Es importante el programa debe seguir la definición del algoritmo, y no al revés.

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. 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@yahoogroups.com[1] usuario y repositorio GitHub.

[1] mailto:UTNFRBAAED@yahoogroups.com

3.3. Productos

- Repositorio público `CppHelloworld`.
- `readme.md`.
- `hello.cpp`
- `output.txt`.

3.4. Entrega

- Mar 30, 13hs.

4

Trabajo #1 — Adición

4.1. Problema

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

4.2. Productos

- Sufijo del nombre de la carpeta: `Adición`
- `readme.md`
- `Adición.cpp`.

4.3. Entrega

- Abr 6, 13hs.

5

Trabajo #2 — Mayor de dos Números

5.1. Problema

Dado dos números informar cuál es el mayor.

5.2. Productos

- Sufijo del nombre de la carpeta: `Mayor`
- `readme.md`.
- `Mayor.cpp`.

5.3. Entrega

- Abr 13, 00hs.

6

Trabajo #3 — Repetición de Frase

6.1. Problema

Enviar una frase a la salida estándar muchas veces.

6.2. Restricciones

Realizar dos versiones del algoritmo y una implementación para cada uno:

- Salto condicional.
- Iterativa esctructurada.

6.3. Productos

- Sufijo del nombre de la carpeta: `Repetición`
- `readme.md` con los dos algoritmos.
- `Saltos.cpp`.
- `Iteración.cpp`.

6.4. Entrega

- Abr 27, 13hs.

7

Trabajo #4 — Ejemplos de Valores y Operaciones de Tipos de Datos

7.1. Tarea

Esta es una *tarea no estructurada*, la cual consiste en escribir un programa que ejemplifique el uso de los tipos de datos básicos de C++ vistos en clase: `bool`, `char`, `unsigned`, `int`, `double`, y `string`.

7.2. Productos

- Sufijo del nombre de la carpeta: `EjemploTipos`
- `EjemploTipos.cpp`.

7.3. Entrega

- May 4, 13hs.