Trabajos de Sintaxis y Semántica de los Lenguajes

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
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 — Conversor de Temperaturas 9
    4.1. Objetivos 9
    4.2. Restricciones 9
    4.3. Entrega 10
5. Léxico de la Calculadora Polaca 11

1

Introducción

Los trabajos prefijados con (@) son comunes para todos los cursos.

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.

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 `CHelloWorld`.
4. Seleccionar, instalar, y configurar un compilador C11.
5. Probar el compilador con un programa `hello.c` que envíe a `stdout` la línea `Hello, World!` o similar.
6. Ejecutar el programa, y capturar su salida en un archivo de texto `output.txt` .
7. Publicar `hello.c` y `output.txt` en GitHub.
8. Escribir y publicar el archivo `readme.md` del proyecto.

9. Enviar a UTNFRBASSL@yahoogroups.com[1] usuario y repositorio GitHub.

[1] mailto:UTNFRBASSL@yahoogroups.com

4

Trabajo #1 — Conversor de Temperaturas

Este trabajo está basado en el ejercicio 1-15 de [K&R1988]:

> 1-15. Reescriba el programa de conversión de temperatura de la sección 1.2 para que use una función de conversión.

4.1. Objetivos

- Realizar el primer trabajo en equipo en el repositorio privado del equipo en **GitHub**.
- Demostrar conocimiento de:
  - Funciones.
  - Archivos header (.h).
  - Interfaces e Implementación.
  - Uso de `make`.

4.2. Restricciones

- Nombre de la carpeta: `Temperatura`.
- Archivos:
  - `FahrCel.c`.

    - `Conversion.h`.
    - `Conversion.c`.
    - `makefile`.
- Utilizar `const`.
- Utilizar `for` con declaración (C99).

4.3. Entrega

- A las 13:00 del día de la sexta clase, Abr 17.

5

Léxico de la Calculadora Polaca

Este trabajo está basado en el la sección 4.3 de [K&R1988]: *Calculadora con notación polaca inversa.*