Laboratorio 2024 de Programación 1

Tarea 1

Información general

Se sugiere leer con mucha atención todo el texto antes de comenzar a trabajar en esta tarea de laboratorio. Es muy importante que se respeten todos los requisitos solicitados en esta sección y las siguientes. Si surgen dudas, pedimos que las formulen en el foro del laboratorio en la página EVA del curso.

Individualidad

Esta tarea se deberá realizar en forma **individual**. Para todas las tareas de laboratorio rige el Reglamento del Instituto de Computación ante Instancias de No Individualidad en los Laboratorios (lectura obligatoria).

Calendario

Entrega: La entrega de la tarea puede realizarse hasta **las 12:00 del 25 de abril**. Los trabajos deberán ser entregados dentro de los plazos establecidos. **No** se aceptarán trabajos fuera de plazo. ***Para poder entregar, el estudiante debe haber realizado anteriormente el Cuestionario 2.***

Reentrega: Todos los estudiantes que realizaron la entrega pueden hacer modificaciones y realizar una segunda entrega (*reentrega*). La reentrega puede realizarse **hasta las 20:00 del 26 de abril.**

Forma de entrega

Se debe entregar un **único** archivo de nombre `tarea1.pas` que debe contener **únicamente** el código de los subprogramas pedidos y eventualmente el de subprogramas auxiliares que se necesiten para implementarlos. La entrega se realiza mediante una actividad en la plataforma EVA en la sección de laboratorio.

Archivos provistos y ejecución de pruebas

El archivo `definiciones.pas` contiene constantes (`SEPARADOR` y `FINALIZADOR`) definidas por los docentes. El programa principal para realizar pruebas es provisto por los docentes en el archivo `principal.pas`. **No se debe modificar ninguno de estos archivos, dado que no serán entregados**. Para usar este programa se debe leer y seguir las instrucciones provistas en la sección "cómo ejecutar los casos de prueba de la Primera Tarea". Los casos de prueba que serán tomados en cuenta para la corrección de la tarea son los que se especifican en esa sección.

Introducción

Al analizar un texto puede resultar de interés computar ciertas métricas (medidas) sobre él. Por ejemplo, pueden resultar medidas interesantes la cantidad de palabras que aparecen en el texto o sus largos. En esta tarea el programa principal procesa un texto con al menos una palabra y obtiene: la cantidad de palabras del texto, el largo de la palabra más larga y el largo de la palabra más corta.

El estudiante debe implementar dos procedimientos, `leerPalabraLargo` y `leerOracionDatos` que serán invocados por el programa principal para leer una oración y calcular las métricas mencionadas.

Subprogramas

Se deben implementar los siguientes subprogramas:

- Procedimiento `leerPalabraLargo`. Lee de la entrada estándar una palabra, es decir, una cadena de caracteres terminada en `SEPARADOR` o `FINALIZADOR`. Retorna en el parámetro de salida `largo` el largo de la palabra (sin contar al `SEPARADOR` o `FINALIZADOR`) y en el parámetro de salida `fin` un booleano que indica si el último carácter es `FINALIZADOR` o no. Asumir que el largo de la palabra es mayor o igual que uno.

```
procedure leerPalabraLargo ( var largo :  integer
                           ; var fin : boolean);
```

- Procedimiento `leerOracionDatos`. Lee de la entrada estándar una oración compuesta de palabras separadas por un único carácter `SEPARADOR`, que finaliza con el carácter `FINALIZADOR`. Retorna en el parámetro de salida `cantPalabras` la cantidad de palabras de la oración, en `masLarga` la cantidad de letras de la palabra más larga y en `masCorta` la cantidad de letras de la palabra más corta. Asumir que la oración tiene al menos una palabra. Notar que para implementar este procedimiento se debe invocar al procedimiento `leerPalabraLargo`.

```
procedure leerOracionDatos (var cantPalabras,
                                masLarga, masCorta : integer);
```

Ejemplos

A continuación se muestran ejemplos de ejecución del programa principal, asumiendo que `SEPARADOR` es `' '` y `FINALIZADOR` es `'.'`.

```
Ingrese la oración: esta es una oracion.
Tiene 4 palabras.
La palabra más larga tiene 7 letras.
La palabra más corta tiene 2 letras.
```

```
Ingrese la oración: sola.
Tiene 1 palabras.
La palabra más larga tiene 4 letras.
La palabra más corta tiene 4 letras.
```

```
Ingrese la oración: oracion con mas palabras y algunas largas como electroencefalografista.
Tiene 9 palabras.
La palabra más larga tiene 23 letras.
La palabra más corta tiene 1 letras.
```

Se pide

Escribir un archivo con todos los subprogramas solicitados. Los encabezados de los subprogramas **deben coincidir exactamente** con los que aparecen en esta letra. Si el estudiante realiza algún cambio se considerará que el subprograma no fue implementado. Si el estudiante lo desea, puede implementar subprogramas auxiliares adicionales (además de los subprogramas pedidos).

Para la corrección, las tareas se compilarán con una versión igual o posterior a **3.0.4 para Linux**. La compilación y la ejecución se realizarán en línea de comandos. El comando de compilación se invocará de la siguiente manera:

```
    fpc -Co -Cr -Miso -gl principal.pas
```

`principal.pas` será el programa principal entregado por el equipo docente.

Debe compilar en la línea de comando. **NO** debe compilar usando ningún IDE.

No está permitido utilizar facilidades de Free Pascal que no forman parte del estándar y no se dan en el curso. Así por ejemplo, no se puede utilizar ninguna de las palabras siguientes: `uses`, `crlscr`, `gotoxy`, `crt`, `readkey`, `longint`, `string`, `break`, `exit`, etcétera.

En esta tarea, como en todos los problemas de este curso, se valorará, además de la lógica correcta, la utilización de un buen estilo de programación de acuerdo a los criterios impartidos en el curso. De esta manera, se hará énfasis en buenas prácticas de programación que lleven a un código legible, bien documentado y mantenible, tales como:

- indentación adecuada,
- utilización correcta y apropiada de las estructuras de control,
- código claro y legible,
- algoritmos razonablemente eficientes,
- utilización de comentarios que documenten y complementen el código,
- utilización de constantes simbólicas,
- nombres mnemotécnicos para variables, constantes, etcétera.

Para resolver la tarea se pueden utilizar todos los conocimientos vistos en el curso.