Laboratorio 2025 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 Foro de consultas Tarea 1 | FING.

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 13:00 del 23 de abril**. Los trabajos deberán ser entregados dentro de los plazos establecidos. **No** se aceptarán trabajos fuera de plazo.

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 24 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

Se proporciona un programa principal, `principal.pas`, para probar los subprogramas solicitados, junto con el archivo `definiciones.pas`, que contiene los tipos definidos necesarios para la tarea. **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".

Introducción

En esta tarea trabajaremos con números naturales positivos (mayores a 0) y exploraremos la relación entre sus dígitos.

Dos números naturales son **anagramas** si uno es la permutación de los dígitos del otro. Por ejemplo, los números 12345 y 35241 son anagramas, mientras que 123 y 4210 no lo son.

Para verificar si dos números son anagramas, implementaremos subprogramas que nos permitan analizar los dígitos de cada número.

Un concepto importante en esta tarea es el **histograma** de un número. Un histograma es una representación de la frecuencia de aparición de cada dígito en un número. Por ejemplo, el histograma del número 111202334 es {0:1, 1:3, 2:2, 3:2, 4:1, 5:0, 6:0, 7:0, 8:0, 9:0}.

Para poder trabajar con números más grandes que los que permite el tipo `integer`, definimos el tipo `Natural` en el archivo `definiciones.pas`. Este tipo debe ser utilizado exclusivamente para representar los números involucrados en la verificación de anagramas. Los estudiantes no deben preocuparse por la definición concreta del tipo `Natural`, pueden tratarlo en forma similar a un entero.

Subprogramas

Se deben implementar los siguientes subprogramas:

- Procedimiento `siguienteDigito`. Dado un número natural positivo `num` (mayor a 0), se extrae el último dígito (el menos significativo) y lo retorna en un parámetro de salida. Modifica el número original eliminando ese último dígito.

  ```
  procedure siguienteDigito(var num: Natural; var digito: integer);
  ```

  Ejemplo de ejecución:

  La llamada:

  ```
  numero := 12345;
  siguienteDigito(numero, digito);
  ```

  produce resultado: `numero = 1234` y `digito = 5`.

- Función `esHistogramaDe`. Recibe un histograma dado en forma de diez enteros que representan la cantidad de ocurrencias de cada dígito (del 0 al 9) y un número natural positivo. La función devuelve `true` si el número tiene exactamente la misma cantidad de cada dígito dado por el histograma, y `false` en caso contrario.

  ```
  function esHistogramaDe(c0, c1, c2, c3, c4,
        c5, c6, c7, c8, c9: integer; num: Natural): boolean;
  ```

  Ejemplos de ejecución:

  La llamada `esHistogramaDe(0, 3, 1, 1, 0, 0, 0, 0, 0, 0, 11321)` devuelve `true`, ya que 11321 tiene exactamente tres ocurrencias de 1, una de 2, una de 3 y cero del resto de los dígitos.

  La llamada `esHistogramaDe(0, 0, 1, 0, 0, 2, 0, 0, 1, 0, 55826)` devuelve `false`, ya que el número tiene ocurrencias del 6 y el histograma no.

  La llamada `esHistogramaDe(0, 3, 1, 0, 0, 2, 1, 0, 1, 0, 55826)` devuelve `false`, ya que el número no tiene ocurrencias del 1 y el histograma sí.

- Función `sonAnagramas`. Determina si dos números naturales positivos son anagramas.

```
function sonAnagramas(num1, num2: Natural): boolean;
```

Ejemplos de ejecución:

```
Ingrese dos números: 12345 35241
Son anagramas.
```

```
Ingrese dos números: 9876 67890
No son anagramas.
```

Se pide

Escribir un archivo `tarea1.pas` 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.