TRABAJO PRÁCTICO 1

FECHA DE ENTREGA LÍMITE: 24/06/19

OBJETIVOS

GENERAL

Aplicar los conocimientos teóricos desarrollados en clases para la implementación de una solución de software que facilite la determinación de un conjunto definido de indicadores basados en la complejidad de múltiples algoritmos.

ESPECÍFICO

Analizar y comparar algoritmos de diferente complejidad en función de posibles caminos de ejecución alternativos.

CONTEXTO

Dado el siguiente **código, parte de un programa en C++:**

```
int a, b;
cout << "Ingrese dos enteros: ";
cin >> a >> b;
if (a >= b) {
  cout << "El máximo es: " << a << endl;
}
else {
  if (b > a) cout << "El máximo es: " << b << endl;
}
```

Luego, la **cantidad de operaciones** involucradas en una ejecución específica depende de los valores ingresados para a y b.

El conjunto de datos de entrada que da lugar a una ejecución se denomina **caso de prueba**. Sean los siguientes conjuntos dos posibles casos de prueba del programa previo:

{ a = 5, b = 8 }

{ a = 15, b = -9 }

Luego, de acuerdo al primer caso se tiene que el total de operaciones ejecutadas es 11, donde:

```
int a, b; ---> 2 operaciones de declaración
cout << "Ingrese dos enteros: "; ---> 1 operación de salida
cin >> a >> b; ---> 2 operaciones de entrada
if (a >= b) {    ---> 2 operaciones de comparación
  cout << "El máximo es: " << a << endl;
}
else {
  if (b > a)    ---> 1 operación de comparación
    cout << "El máximo es: " << b << endl; ---> 3 operaciones de
salida
}
```

En cambio, si se analiza el segundo caso de prueba se tiene que el total de operaciones ejecutadas es 9. Esto se justifica ya que:

```
int a, b; ---> 2 operaciones de declaración
cout << "Ingrese dos enteros: "; ---> 1 operación de salida
cin >> a >> b; ---> 2 operaciones de entrada
```

```
if (a >= b) {    ---> 1 operación de comparación
  cout << "El máximo es: " << a << endl;    ---> 3 operaciones de salida
}
else {
  if (b > a)
    cout << "El máximo es: " << b << endl;
}
```

Luego, la cantidad de operaciones a ejecutar varía en función de las **operaciones (aritméticas y lógicas)** asociadas a cada camino posible del algoritmo bajo estudio.

APLICACIÓN DE SOFTWARE A DESARROLLAR

Tomando en consideración un conjunto de algoritmos predefinidos por la cátedra, se debe implementar una aplicación C++ que genere un archivo con la salida de los casos de prueba procesados y visualizar por pantalla una tabla comparativa de los múltiples casos de prueba y sus salidas.

MATERIAL QUE ACOMPAÑA A ESTE ENUNCIADO

La cátedra provee:

- Los *algoritmos* C++ a ser analizados (**algoritmoN.cpp**), donde N identifica al número de algoritmo que contiene.
- Para cada algoritmo, un conjunto (no exhaustivo) de *casos de prueba* que facilitan su evaluación (**pruebaAlgoritmoN.txt**). Cada archivo inicia con la cantidad de casos de prueba que incluye y, luego, cada línea corresponde a un caso de prueba. Si un caso de prueba requiere más de un dato, los datos se encuentran separados por espacios en blanco.

- La *salida esperada* por ejecución de cada caso de prueba entregado (**ejecucionAlgoritmoN.txt**).

FUNCIONAMIENTO DE LA APLICACIÓN

Al iniciar la aplicación el usuario verá un menú similar al siguiente:

```
            Menú Principal
          ========================
          a.- Analizar Algoritmos
          b.- Visualizar Resultados
          s.- Cerrar la aplicación
```

A continuación, el usuario debe ingresar la opción que desea ejecutar. En el caso de seleccionar la opción “Visualizar Resultados” se debe tener en cuenta que previamente debe haberse realizado el análisis de alguno de los algoritmos disponibles en el menú “Analizar Algoritmos”. En caso de que no existan resultados para visualizar, se debe indicar esta situación al usuario y la aplicación debe regresar al menú principal.

Opción “Analizar Algoritmos”

Se presenta al usuario un listado con los algoritmos disponibles para su análisis y, luego, el programa queda a la espera de que el usuario indique cuál es la opción del menú que desea ejecutar.

```
            Analizar Algoritmos C++
          ==========================
    a.- Algoritmo #1 SECUENCIAL (Solución a problema de sumatoria)
    b.- Algoritmo #2 IF/ELSE (Solución a problema de selección completa)
    c.- Algoritmo #3 SWITCH (Solución a problema de selección múltiple)
    d.- Algoritmo #4 FOR (Solución a problema de sumatoria)
    e.- Algoritmo #5 FOR (Solución a problema repetitivo-anidado)
    f.- Algoritmo #6 DO-WHILE / IF / SWITCH (Solución problema zodíaco)
    g.- Algoritmo #7 WHILE (Solución problema decimal-binario-octal)
    s.- Volver al Menú Anterior
```

Cuando el usuario ingresa una opción válida, se procede a ejecutar el análisis de los casos de prueba incluidos en el archivo **pruebaAlgoritmoN.txt** sobre el algoritmo elegido. Una vez ejecutado el análisis de un algoritmo, no podrá repetirse (es decir, en una ejecución de la aplicación un mismo algoritmo no puede analizarse más de una vez). En caso de que el usuario intente ejecutar nuevamente el análisis, se le informa el error y el programa vuelve al menú "Analizar Algoritmos".

Como resultado del análisis se genera un archivo de salida **ejecucionAlgoritmoN.txt** que contiene, para cada caso de prueba analizado:

- Cantidad de operaciones involucradas en su ejecución.
- Cantidad de variables declaradas.
- Tamaño total (en bytes) ocupado por las variables.

Además, se debe mostrar por pantalla claramente el caso de prueba analizado y el resultado de su ejecución (esto es, las salidas generadas en el algoritmo original como resultado de cada uno de los casos de prueba ejecutados).

Una vez finalizado el análisis, se retorna al menú "Analizar Algoritmos" a fin de continuar realizando ejecuciones de prueba sobre los algoritmos restantes.

Operaciones a Considerar en el Análisis

La solución propuesta debe contemplar:

- Declaración de variables.
- Operaciones de entrada y salida.
- Inicialización, asignaciones y operadores abreviados.
- Conversión de tipos de datos (automática, implícita y explícita).
- Operadores aritméticos, relacionales, y lógicos.

- Funcionamiento de las estructuras de control secuenciales, de selección y de repetición.
- Evaluación por cortocircuito.

Opción "Visualizar Resultados"

Se presenta al usuario un listado con los algoritmos disponibles para visualizar el resultado del análisis:

```
                Visualizar Resultados Algoritmos C++
              ========================================
    a.- Algoritmo #1 SECUENCIAL (Solución a problema de sumatoria)
    b.- Algoritmo #2 IF/ELSE (Solución a problema de selección completa)
    c.- Algoritmo #3 SWITCH (Solución a problema de selección múltiple)
    d.- Algoritmo #4 FOR (Solución a problema de sumatoria)
    e.- Algoritmo #5 FOR (Solución a problema repetitivo-anidado)
    f.- Algoritmo #6 DO-WHILE / IF / SWITCH (Solución problema zodíaco)
    g.- Algoritmo #7 WHILE (Solución problema decimal-binario-octal)
    s.- Volver al Menú Anterior
```

Cuando el usuario ingresa una opción válida y el algoritmo elegido fue analizado, se debe procesar el archivo **ejecucionAlgoritmoN.txt** para luego mostrar por pantalla un informe tabular con el formato que se presenta a continuación (una línea por cada caso de prueba).

```
-------------------------------------------------------------------------------
|Nro Caso de prueba | Cant. Operaciones | Cant. Variables | Tam. (bytes) Variables |
-------------------------------------------------------------------------------
|                   |                   |                 |                        |
-------------------------------------------------------------------------------
|                   |                   |                 |                        |
-------------------------------------------------------------------------------
```

Si el algoritmo elegido no fue analizado, se debe mostrar un mensaje por pantalla indicando la situación y luego volver al menú de "Visualizar Resultados".

Opción “Cerrar la Aplicación”

Se le pide al usuario que confirme la acción. En caso afirmativo, se cierra la aplicación. En otro caso, se retorna al menú inicial.

AMIGABILIDAD, VALIDACIONES E INTERFAZ

Los lineamientos establecidos en este documento deben respetarse para el desarrollo de la aplicación solicitada. Se deberá garantizar una correcta validación de los datos ingresados por el usuario, como así también la presencia de mensajes claros e intuitivos.

Se valorará positivamente que los grupos formulen soluciones que, respetando los lineamientos establecidos, mejoren la interfaz gráfica incorporando pantalla de inicio, menú de ayuda, mensajes expresivos, sonidos, etc. Además, se podrá considerar las relaciones entre opciones a fin de mantenerlas habilitadas/deshabilitadas.

EJEMPLO

Supongamos que se quiere analizar el algoritmo IF/ELSE (presentado en el apartado “Contexto”) de acuerdo al archivo *pruebaAlgoritmo0.txt*.

```
 pruebaAlgoritmo0.txt
----------------------
 2
 5 8
 15 -9
```

Luego, el contenido del archivo *ejecucionAlgoritmo0.txt* será el que se visualiza a continuación y la salida en pantalla deberá ser similar a la que se muestra a continuación.

```
 ejecucionAlgoritmo0.txt
-------------------------
```

```
 11 2 8
 9 2 8
 EOF
```

```
 SALIDA CASO DE PRUEBA (5,8)
------------------------------
El máximo es: 8.

 SALIDA CASO DE PRUEBA (15,-9)
------------------------------
El máximo es: 15.
```

Al ingresar a la opción de visualización de este algoritmo, el informe tabular quedaría así:

```
-------------------------------------------------------------------------------------
|Nro Caso de prueba | Cant. Operaciones | Cant. Variables | Tam. (bytes) Variables |
-------------------------------------------------------------------------------------
|                 1|                11|                2|                      8|
-------------------------------------------------------------------------------------
|                 2|                 9|                2|                      8|
-------------------------------------------------------------------------------------
```

Si, en cambio, el contenido del archivo *pruebaAlgoritmo0.txt* fuera el siguiente:

```
 pruebaAlgoritmo0.txt
----------------------
 5
 15 9
 15 -9
 2 0
 10 2
 63 -96
```

El contenido del archivo *ejecucionAlgoritmo0.txt* se corresponderá con el esquema que se presenta a continuación y la salida por pantalla será similar a la siguiente.

```
 ejecucionAlgoritmo0.txt
-------------------------
 9 2 8
 9 2 8
 9 2 8
 9 2 8
 9 2 8
 EOF
```

```
 SALIDA CASO DE PRUEBA (15,9)
------------------------------
El máximo es: 15.

 SALIDA CASO DE PRUEBA (15,-9)
-------------------------------
El máximo es: 15.

 SALIDA CASO DE PRUEBA (2,0)
-------------------------------
El máximo es: 2.

 SALIDA CASO DE PRUEBA (10,2)
-------------------------------
El máximo es: 10.

 SALIDA CASO DE PRUEBA (63,-96)
-------------------------------
El máximo es: 63.
```

Al ingresar a la opción de visualización de este algoritmo, el informe tabular quedaría así:

```
 ---------------------------------------------------------------------------------------
|Nro Caso de prueba | Cant. Operaciones | Cant. Variables | Tam. (bytes) Variables |
---------------------------------------------------------------------------------------
|                 1|                 9|               2|                       8|
```

```
-------------------------------------------------------------------------------
|                2|                9|                2|                8|
-------------------------------------------------------------------------------
y así hasta llegar al 5to caso de prueba.
```

Luego, se debe elaborar una solución genérica que realice el cálculo de las operaciones de acuerdo al camino que se formula cuando se ejecuta cada uno de los casos de prueba incluidos en el archivo pruebaAlgoritmoN.txt.

PAUTAS DE ENTREGA

CONFORMACIÓN DE GRUPOS

El trabajo debe ser realizado en grupos de 3 alumnos (de la misma o de distinta comisión).

CALIFICACIÓN

La solución entregada por cada grupo será sometida a un conjunto de casos de prueba (distinto) de los entregados a los alumnos. En función de su rendimiento y de la eficiencia de la solución desarrollada, cada grupo recibirá una calificación numérica que se computará de la misma forma que los exámenes parciales en cuanto a la regularidad/promoción de la materia.

Los trabajos no entregados en tiempo y forma, no serán evaluados y se considerarán *no aprobados*. Lo mismo ocurrirá con los trabajos entregados por grupos que no respeten la conformación de grupos (3 alumnos).

POLÍTICA ANTIPLAGIO

Las entregas realizadas por los grupos son individuales. Los archivos entregados serán analizados con herramientas de software específicas para detectar plagios de código.

Los trabajos que no cumplan con las pautas de autoría establecidas, no serán evaluados y se considerarán *no aprobados*.

DOCUMENTACIÓN A ENTREGAR

Las entregas se realizarán exclusivamente por medio de la tarea TP1 AEDD – Entrega creada para tal fin en el campus virtual de la materia. Los archivos asociados a cada entrega deben comprimirse en un archivo ZIP/RAR nombrado bajo la estructura:

AEDD_GRUPO_apellido1_apellido2_apellido3

Cada entrega la realizará sólo uno de los integrantes del grupo.
Deberán entregar a través de la tarea **TP1 AEDD – Entrega** los siguientes archivos dentro del zip:

- **Integrantes.txt:** archivo de texto que contendrá el apellido, nombre y correo electrónico de cada uno de los integrantes del grupo.
- El **programa C++** (cpp) y su respectivo **archivo ejecutable** (exe).
- Una **copia comentada** (de manera similar al ejemplo dado) de cada uno de los archivos **algoritmoN.cpp** indicando en cada línea de código la operación que se contabiliza, en cuánto y por qué.
  Por ejemplo: `cin >> a >> b; // 2 operaciones de entrada porque ingresan dos variables por teclado.`