Ingeniería de Software I
Práctica #3: Variables Globales y Variables Locales

Objetivo

Ejercer la programación modular abstrayendo la implementación de subrutinas de tipo procedimiento y de tipo función y al mismo tiempo distinguir la diferencia entre el uso adecuado de variables de ámbito local apoyado del paso de parámetros y los casos excepcionales de variables de ámbito global.

Descripción

La práctica trata sobre la modularización del código fuente implementado para la práctica #2, logrando que esta práctica #3 funcione al exterior igual que como se especificó para la #2 (refiriéndonos a ello como *requerimientos funcionales*), pero mejorando su implementación al interior según requerimientos de esta práctica (a citar como *requerimientos no funcionales*). Además, para todas las variables locales utilizadas dentro de la subrutina main de la práctica #2, que se pueden considerar como de ámbito "global" para todos los requerimientos de esa práctica #2, para esta práctica #3 se decidirá cuales declararlas como de ámbito global y cuáles serán de ámbito local acorde a las subrutinas diseñadas.

Como detalle adicional, la práctica ha de determinar el porcentaje de ISR a aplicar acorde al siguiente tabulador de ingresos:

Límite inferior	Límite superior	Porcentaje ISR
0	10000.00	11%
10000.01	20000.00	15%
20000.01	En adelante	20%

El anterior tabulador se aplica a la *ganancia bruta*, es decir el *ingreso* menos el *gasto*.

Ejemplos:

1) Para un ingreso anual de 10000 y un gasto anual de 500, donde los 10000 y los 500 resulten de la suma de los ingresos y gastos de todos los meses:

```
Cálculo de impuestos anual
***Tabla Ingresos y Retenciones***
Ingresos                10000.00
(+) IVA                  1600.00
(=) Subtotal            11600.00
(-) Retención ISR        1000.00
(-) Retención IVA        1000.00
(=) Total                9600.00
***Tabla Ganancias***
Ingresos                10000.00
(-) Gastos                500.00
(=) Ganancia Bruta       9500.00
(-) ISR 11.00%           1045.00
(=) Ganancia Neta        8455.00
***Tabla ISR***
ISR 11.00%               1045.00
(-) ISR Retenido         1000.00
(=) ISR a Pagar            45.00
***Tabla IVA***
IVA                      1600.00
(-) Gastos IVA             80.00
(-) Retención IVA        1000.00
(=) IVA a Pagar           520.00
Presione entrar para continuar . . .
```

2) Para un ingreso anual de 20000 y un gasto anual de 2000, donde los 20000 y los 2000 resulten de la suma de los ingresos y gastos de todos los meses:

```
Cálculo de impuestos anual
***Tabla Ingresos y Retenciones***
Ingresos                20000.00
(+) IVA                  3200.00
(=) Subtotal            23200.00
(-) Retención ISR        2000.00
(-) Retención IVA        2000.00
(=) Total               19200.00
***Tabla Ganancias***
Ingresos                20000.00
(-) Gastos               2000.00
(=) Ganancia Bruta      18000.00
(-) ISR 15.00%           2700.00
(=) Ganancia Neta       15300.00
***Tabla ISR***
ISR 15.00%               2700.00
(-) ISR Retenido         2000.00
(=) ISR a Pagar           700.00
***Tabla IVA***
IVA                      3200.00
(-) Gastos IVA            320.00
(-) Retención IVA        2000.00
(=) IVA a Pagar           880.00
Presione entrar para continuar . . .
```

3) Para un ingreso anual de 30000 y un gasto anual de 4000, donde los 30000 y los 4000 resulten de la suma de los ingresos y gastos de todos los meses:

```
Cálculo de impuestos anual
***Tabla Ingresos y Retenciones***
Ingresos                30000.00
(+) IVA                  4800.00
(=) Subtotal            34800.00
(-) Retención ISR        3000.00
(-) Retención IVA        3000.00
(=) Total               28800.00
***Tabla Ganancias***
Ingresos                30000.00
(-) Gastos               4000.00
(=) Ganancia Bruta      26000.00
(-) ISR 20.00%           5200.00
(=) Ganancia Neta       20800.00
***Tabla ISR***
ISR 20.00%               5200.00
(-) ISR Retenido         3000.00
(=) ISR a Pagar          2200.00
***Tabla IVA***
IVA                      4800.00
(-) Gastos IVA            640.00
(-) Retención IVA        3000.00
(=) IVA a Pagar          1160.00
Presione entrar para continuar . . .
```

Requerimientos Funcionales

1. Entregar archivo(s) fuente para aplicación de consola que cumpla(n) con la descripción citada y los siguientes requerimientos.
2. Imprimir en consola el título de la aplicación y a continuación el menú siguiente:

```
CÁLCULO DE IMPUESTOS ANUAL

Menú principal:
1. Establecer mes para captura (mes actual es Enero)
2. Captura de ingresos
3. Captura de gastos
4. Mostrar lista de ingresos anual
5. Mostrar lista de gastos anual
6. Cálculo de impuestos anual
7. Guardar en archivo
8. Salir

Opción:
```

3. Contemplar para esta práctica las descripciones citadas en prácticas #1 y #2.
4. Cumplir con las impresiones de consola mostradas en la práctica #2, salvo los cambios necesarios para cumplir con los requerimientos siguientes.
5. Para cada dato requerido del usuario imprimir la solicitud de información correspondiente (printf, cout, etc.) y a continuación esperar por la entrada de datos (scanf, cin, etc.).
6. Se realice un mostrado cíclico del menú mientras no se elija la opción “Salir” de dicho menú.
7. La opción #1 del menú permita establecer el *mes de trabajo* a considerar para las opciones #2 y #3 del menú, mostrando a Enero como el mes #1.
8. Validar que el mes de trabajo siempre sea un mes válido.
9. Al mostrar el menú, para la opción #1 se muestre el mes de Enero como mes a considerar para las opciones #2 y #3 del menú, y cada que cambie el mes, se muestre el *mes de trabajo* seleccionado.
10. La opción #2 del menú permita capturar el ingreso de sólo el *mes de trabajo*, reemplazando el monto anterior en el mes por otro monto (inclusive reemplazar por un monto inferior).
11. La opción #3 del menú permita capturar el gasto de sólo el *mes de trabajo*, reemplazando el monto anterior en el mes por otro monto (inclusive reemplazar por un monto inferior).
12. Se evite la captura de ingreso menor a cero o gasto menor a cero, solicitando en dado caso de nuevo el monto.
13. La opción #4 muestre la lista de ingresos de todos los meses del año.
14. La opción #5 muestre la lista de gastos de todos los meses del año.
15. Para cada ingreso o gasto no definido por el usuario se muestre un valor de cero.
16. Al elegir la opción #6, sumar todos los ingresos y todos los gastos, e imprimir en la consola las 4 tablas citadas en la descripción de la práctica #1, de forma que sus

contenidos sean claros a la vista y los cálculos sean correctos para cualesquier sumatoria de ingresos y gastos.
17. Todos los valores capturados y resultados impresos contemplen centavos; al imprimir los resultados estos se muestren alineados por el punto decimal (esto último hacerlo al menos en Lenguajes C/C++).
18. En la impresión del cálculo de impuestos (opción #6 del menú) mostrar el porcentaje de ISR que corresponda.
19. Si el usuario elige una opción no contemplada en el menú se imprima “Opción no válida”.
20. Al terminar de ejecutar una opción del menú (incluso una no válida), excepto la opción “Salir”, mostrar el mensaje “Presione entrar para continuar. . .” y esperar por la tecla “entrar”.
21. Al elegir la opción “Salir” del menú se termine la aplicación sin pausa alguna.
22. Para poder visualizar todo lo impreso por la aplicación, limpiar la pantalla e imprimir de tal manera que no se requiera redimensionar el tamaño por defecto de la consola (25 filas por 80 columnas), ni usar las barras de desplazamiento.
23. Los listados de ingresos, gastos y el reporte de cálculo de impuestos se muestren alineados para la impresión de montos respecto al punto decimal para los centavos.
24. Para toda opción del menú principal que no sea implementada, al ser elegida por el usuario mostrar el mensaje “En construcción =)”. Por ejemplo, si el(la) alumn@ no implementa el comportamiento de la opción "7. Guardar en archivo", al elegir dicha opción se imprima en pantalla "En construcción =)".

Requerimientos No Funcionales

25. Declarar y definir a continuación de la inclusión de librerías todas las constantes, o sea, aquellas relativas a los porcentajes citados en la descripción de la práctica #1 y demás constantes que se necesiten
26. A continuación de las constantes, declarar las variables globales (ó variables de programa), de uso interno del programa a utilizar por más de una subrutina, es decir, aquellas dedicadas a almacenar el estado actual del programa ante el usuario.
27. Implementar una *subrutina de inicialización* (procedimiento o función), sin parámetros para inicializar todas las variables globales del programa.
28. Implementar una subrutina (procedimiento o función) sin parámetros, para mostrar el menú.
29. Definir en tiempo de compilación una estructura arreglo de cadenas para los nombres de los 12 meses del año.
30. Se utilice una estructura de datos arreglo vector, para almacenar en ella los *ingresos* de cada uno de los meses del año; de la misma forma sea para los *gastos* mensuales; ambos arreglos sean independientes.
31. Se utilice la estructura de control “do-while” para efectuar el mostrado cíclico del menú.
32. La estructura de control “do-while” del punto anterior utilice una variable estilo bandera.

33. Para la “pausa” (espera de tecla entrar) a realizar después de mostrar los resultados de las opciones 1 a la 7 del menú utilizar la bandera del punto anterior para que cuando la bandera indique que se volverá a mostrar el menú, se realice dicha pausa.
34. La subrutina main (programa principal) incluya en código fuente la estructura de control do-while mencionada previamente.
35. Para cada opción del menú excepto “Salir” se implemente un procedimiento sin parámetros encargado de:
    1) Mostrar un título correspondiente a la opción seleccionada;
    2) hacer las peticiones de datos al usuario según la práctica #2;
    3) realizar la(s) operación(es) correspondiente(s), y si aplica, llamar a la(s) función(es) ó procedimiento(s) necesarios;
    4) finalmente imprimir en consola el resultado de la operación.
36. Para cada opción elegible en el programa principal, solo se instruya la llamada a la subrutina que corresponda, es decir, haya solo una proposición por cada case, además del break.
37. Se utilice un arreglo vector para almacenar los porcentajes de ISR citados en la descripción.
38. Se utilice un arreglo matriz ó un vector para relacionar los rangos de montos con los porcentajes de ISR del punto anterior, mediante una de las siguientes formas:

matrizMontos

Fila/Col	0	1
0	0	10000.00
1	10000.01	20000.00
2	20000.01	En adelante

porcentajesISR

Fila/Col	0
0	11
1	15
2	20

…o bien

vectorMontos

Fila/Col	0
0	0
1	10000.01
2	20000.01

porcentajesISR

Fila/Col	0
0	11
1	15
2	20

39. Las estructuras de datos de los dos puntos anteriores sean inicializadas dentro de la *subrutina de inicialización* citada anteriormente, o bien, al declarar las variables.
40. Utilizar comentarios para mostrar las siguientes elementos del programa:
    a) Declaración y definición de constantes;
    b) Variables globales de uso interno del programa;
    c) Variables de programa modificables por el usuario.
41. Declarar variables para cada uno de los conceptos citados en la práctica #1, excepto *ingreso* y *gasto.*
42. Al elegir la opción #6, definir una variable *ingresoTotal* en base a la suma acumulada de los ingresos mensuales; de la misma forma sea para una variable *gastoTotal.*
43. Al elegir la opción #6, definir las demás variables según práctica #1 en base las variables *ingresoTotal* y *gastoTotal.*

44. Implementar una función que reciba un parámetro de tipo flotante como monto y regrese el porcentaje aplicable de ISR correspondiente al rango en que se ubique dicho monto. Devolver 0 a 100 (porcentajes enteros). En caso de un monto negativo, regrese porcentaje cero.
45. Llamar a la función del punto anterior dándole como argumento la "gananciaBruta", pues para determinar el ISR aplicable se usa este monto y no los ingresos iniciales.
46. Implementar la función del punto anterior con un diseño flexible, es decir, que funcione sin cambio alguno sea cual fuere el tamaño presente o futuro del vector de porcentajes.
47. Evitar el uso de operadores dentro de toda sentencia usada para salida de datos.
48. La opción "Salir" del menú funcione como se citó en la descripción, evitando usar la sentencia return o exit dentro del bucle del programa principal.
49. Todas las constantes necesarias para cumplir con los requerimientos previamente citados, sean de tipo entero.
50. Para la subrutina del menú #6, escribir comentarios en el código fuente destacando cada una de las siguientes partes del programa:
    a) Declaración de variables;
    b) Cálculo de impuestos;
    c) Salida de datos.
51. Las partes del programa citadas anteriormente deben ser exclusivas, es decir, ningún cómputo (del cálculo de impuestos) debe llevarse a cabo en la sección de entrada de datos (captura de información) ni en la sección de salida de datos (presentación de resultados).
52. Declarar los prototipos para cada una de las subrutinas en este programa.

Requerimientos de Archivo Funcionales

53. Cumplir con los requerimientos de archivo de la práctica anterior.
54. Los listados de ingresos y gastos se guarden en archivos separados, con nombres claros y significativos para cada uno de los archivos.
55. Al elegir la opción "Guardar en archivo", los listados de ingresos y gastos se guarden en medio secundario.
56. Si al iniciar la aplicación, los listados de ingresos y gastos ya se habían guardado en medio secundario, inicializar las listas de ingresos y gastos con la información guardada.
57. En el archivo de ingresos, usar una línea de texto para almacenar el ingreso de cada uno de los meses; esto es, el archivo contenga doce líneas de texto, una por cada mes; no escribir los nombres de los meses en el archivo, sino solo los montos correspondientes a cada mes del año fiscal.
58. Análogamente al punto anterior, hágase también la escritura en el archivo de gastos.

Requerimientos de Archivo No Funcionales

59. Al iniciar el programa, cargar ambos listados durante la ejecución de la subrutina de inicialización, citada previamente en los requerimientos no funcionales.

Requerimientos No Funcionales de Bases de Datos

60. Incluir dos carpetas nombradas “Variante1” y “Variante2”, donde “Variante1” contenga un programa basado en archivos acorde a todos los requerimientos citados previamente.
61. Incluir en la carpeta “Variante2”, una segunda versión del programa elaborado acorde a los requerimientos citados previamente, modificando todas aquellas subrutinas relativas a escribir la datos en archivos de texto, para que en su lugar se almacenen los datos en una base de datos.
62. Al iniciar el programa, cargar desde la base de datos ambos listados de ingresos y gastos durante la ejecución de la subrutina de inicialización, citada previamente en los requerimientos no funcionales.
63. Entregar un programa de aplicación de consola por separado del programa elaborado previamente, el cual se encargue de crear todas las tablas de base de datos necesarias para cumplir con los requisitos previos.
64. El gestor de base de datos (SGBD) aplicable para el funcionamiento en ejecución del software elaborado sea MySQL.

Requerimientos para Puntos Extras

65. Entregar en el mismo comprimido carpetas diferentes para cada lenguaje:
    a) Una nombrada “C” con los códigos fuente en C (ANSI);
    b) Una nombrada “C++” con los códigos fuente en C++;
    c) Una nombrada “Java” con los códigos fuente en Java;
    d) Una nombrada “C#” con los códigos fuente en C#;
    e) Una nombrada “English” con los códigos fuente escritos en su totalidad en inglés (solo para el lenguaje de programación de su preferencia).

    …las primeras opciones se redacten absolutamente en español y la última absolutamente en inglés. Si incluye 2 carpetas con lenguajes OO, uno de ellos puede redactarse en inglés y sin necesidad de entregar la versión en español para ese lenguaje.
66. Para cada lenguaje cumplir con los requerimientos de archivo.

Criterios de Evaluación

- Los establecidos en las “Reglas de Operación y Evaluación” del curso.
- Cumplir con la fecha límite de entrega citada en el Excel de Actividades.
- Calificación en base a cobertura de requerimientos.
- Cumplir con Requerimientos de Valor Agregado en Código Fuente (hasta el req. “JJ”)
- Entrega en al menos un lenguaje: C++ (ANSI) o Java o C#.
- Es indispensable la entrega de un programa con variables, comentarios e impresiones a consola completamente en idioma Español.
- Entregar el programa preferentemente después de haber recibido retroalimentación de la práctica anterior, ya sea a través del Excel Evaluación o de sus compañer@s de equipo; lo anterior dado que para la evaluación, todo requerimiento de la práctica anterior se cuenta como un requerimiento más de esta práctica.