Guía de Trabajos Prácticos

Esta guía contiene los Trabajos Prácticos (TP) obligatorios del curso R1043, incluyendo una guía para su entrega con el formato de presentación y un cronograma con las fechas de entrega.
Durante el transcurso del año de deberán entregar siete Trabajos Prácticos, que se caracterizan por ser incrementales. Esto significa que en cada TP se irán introduciendo funcionalidades y mejoras respecto del TP anterior. A su vez, se deberán utilizar las herramientas SVN, Makefile y Doxygen cuando se lo indique.

Formato de presentación

- Los archivos fuentes deben tener en todos los casos los comentarios necesarios para clarificar su lectura.

- Cada subrutina/función, debe contar con un encabezado describiendo la operación que realiza, los parámetros que espera como entrada, y los resultados que debe presentar, indicando formato y método de entrega.

- Como encabezado del programa, debe haber un comentario que explique claramente que hace dicho programa, y las instrucciones detalladas (comandos) para su compilación y "linkeo".

- Cada uno de los trabajos prácticos será entregado a través del SVN. Para ellos tener en cuenta las indicaciones descriptas en el Apéndice I que se encuentra al final de la guía.

- TODOS los ejercicios son obligatorios.

- **La no entrega de un trabajo práctico en la fecha establecida equivale a considerar un ausente. En consecuencia se considerará No Aprobado dicho práctico. De acuerdo con el reglamento vigente, la aprobación de los Trabajos Prácticos requiere el 80% de los Prácticos Aprobados. En el caso de esta guía de Trabajos Prácticos se requiere la aprobación del 80% de los estipulados.**

Cronograma de Entregas y Herramientas a Incluir

	TP Nº 1	TP Nº 2	TP Nº 3	TP Nº 4	TP Nº 5	TP Nº 6	TP Nº 7
Fecha de Entrega	06/05/14	20/05/14	03/06/14	05/08/14	20/08/14	02/09/14	23/09/14
SVN	X	X	X	X	X	X	X
Makefile		X	X	X	X	X	X
Doxygen			X	X	X	X	X
Librería con las funcioes				X	X	X	X
Conocimientos Necesarios para su Resolución	Sentencias básicas de C	C básico, math.h, funciones	Enum, Strings	Arrays, Punteros, Punteros a Función, Memoria dinámica	Matrices, Estructuras	Archivos	Sockets

Trabajo Práctico Nº 1

Elabore un programa que permita realizar operaciones matemáticas entre dos operandos. Para ello se ingresan dos valores reales y el símbolo de la operación ('+', '-', '*', '/','%', '^'). Se deberán presentar en pantalla los datos ingresados, la operación y el resultado. Si el símbolo utilizado no correspondiera a ninguna de las cuatro operaciones deberá presentar un mensaje de “Operación NO Válida”. A su vez, si la operación es de división y el valor ingresado como divisor es 0 (cero), presentará por pantalla un mensaje de “Operación NO Válida División por Cero”. (El programa deberá resolverse mediante el uso de la estructura switch). La implementación de la operación de potencia ('^') debe realizarse con un código propio.

Trabajo Práctico Nº 2

Modifique el programa del Trabajo Práctico Nº 1 para incluir nuevas funcionalidades. Agregue un menú inicial que permita realizar los siguientes cálculos:

1. Operaciones con dos operandos y un operador entre ellos (las realizadas en el TP Nº 1)
2. Operación de factorial. Implemente una función para su cálculo.
3. Cálculo de la función exponencial (e^x) para un exponente ingresado. Para la obtención del número de Euler se debe tener en cuenta que se puede obtener un valor aproximado del número de Euler con la siguiente fórmula:

   $$e = 1 + 1/1! + 1/2! + 1/3! + 1/4! + 1/5! + ...$$

   Implemente dicha aproximación con un ciclo repetitivo de manera que termine cuando la diferencia entre dos aproximaciones sucesivas difiera en menos de 10^{-9}.
4. Cálculos trigonométricos (sin, cos, tan, asin, acos, atan). Los ángulos se ingresarán en grados. Para el cálculo del arco tangente, implemente una segunda función llamada atan2 que calcule el arco tangente del cociente de dos parámetros recibidos y como resultado obtenga un ángulo entre 0º y 360º.
5. Conversión de número decimal a binario y/o hexadecimal. Para este punto se requiere utilizar funciones en lugar de los modificadores de printf.

Trabajo Práctico Nº 3

Modifique el programa del Trabajo Práctico Nº 2 unificando los items del 1 al 4 en uno único que permita ingresar un cálculo genérico por medio de un string. El mismo permite el ingreso de múltiples términos, la utilización de paréntesis y todas las funciones admitidas en el TP Nº 2. Por ejemplo, se le podrá ingresar la siguiente operación:

12.3*e^(0.5*3)*sin(12*3+0.3)

Como guía tenga en cuenta que:

Será conveniente trabajar con strings hasta obtener el resultado final. En la resolución primero deberá identificar el primer paréntesis a resolver (de no haber ningún paréntesis se debe resolver siguiéndose las reglas de prioridad de operaciones del álgebra); a continuación resolver las funciones fundamentales (potencias y trigonométricas); luego, separar en términos para resolver cada uno de ellos; y finalmente, realizar la suma de términos.

- En cada uno de los pasos el contenido resuelto se devolverá en un string que servirá para reemplazar la operación evaluada dentro de la fórmula original.
- Para procesar los paréntesis, desarrolle una función que responda al siguiente prototipo:

  Void ObtenerInteriorParentesis(char* strFormula, char* strIntPar);

  donde ***strFormula*** es el puntero a char que referencia al string leído desde teclado con la fórmula completa, y ***strIntPar*** es el puntero a char que referencia al string que encierra la fórmula parcial entre paréntesis.

  Tenga en cuenta que el primero a resolver será cuando encuentre una apertura de paréntesis y luego un cierre.
- Desarrolle una función que reemplace una sección de un string (una operación) por otra (su resultado), permitiendo que su tamaño sea diferente:

  void ReemplazaSubString(char* strOriginal, char* strReemplazado, char* strReemplazo)

  donde ***strOriginal*** es el puntero a char que referencia al string completo, s***trReemplazado*** es el puntero a char que referencia a la cadena de caracteres a ser reemplazada y **strReemplazo** es el puntero a char que referencia al string de reemplazo.

Otras funciones de utilidad podrían:

- Verificar que la cantidad de paréntesis de apertura y cierre sean iguales.
- Buscar un caracter particular dentro de un string. De esta forma podrá identificar los operadores y funciones admitidas.
- Resolver los paréntesis de forma recursiva hasta eliminarlos en su totalidad.

Trabajo Práctico Nº 4

Agregue a la aplicación que se viene desarrollando en esta guía de trabajos prácticos, la posibilidad de evaluar funciones predefinidas en un rango de valores, así como calcular su derivada e integral. Para ello agregue al menú principal de la aplicación esta nueva opción.
Una vez ingresado a la misma, se le presentará al usuario la lista de funciones disponibles (propuestas por el alumno) y se solicitará el ingreso del rango de valores de x en el cual se evaluará la función f(x), así como la separación entre puntos (xmin, xmax y deltax). Llamamos evaluar la función a obtener sus valores de salida para los valores de entrada x definidos por el usuario.
Para la derivada utilizar una aproximación por diferencias finitas de primer o segundo orden (elija una, es opcional realizar ambas). Para el cálculo de la integral utilizar el método del trapecio o el de Simpson (elija uno, es opcional realizar ambos).
Tanto la función de evaluación, como la de derivada y la de integral deberán ser implementadas de forma genérica mediante un puntero a función. Dichas funciones recibirán un puntero a un array con los valores de x a utilizar, un puntero a la función que se desea evaluar y un puntero al array de salida donde se guardará el resultado.

```
void EvaluaFuncion(float* x, (float) (*funcion)(float), float* y);
void DerivadaFuncion(float* x, (float) (*funcion)(float), float* y);
void IntegralFuncion(float* x, (float) (*funcion)(float), float* y);
```

Recomendación: Realice el pedido de memoria al sistema antes de invocar a estas funciones, liberando la misma luego de retornar desde la función y una vez concluido su tratamiento.

Trabajo Práctico Nº 5

Agregue a la aplicación que se viene desarrollando en esta guía de trabajos prácticos una opción para realizar operaciones entre matrices. Las operaciones a implementar son la suma, la resta y la multiplicación.
La aplicación pedirá que se ingrese cada una de las matrices de a una por vez, utilizando corchetes para determinar el inicio y fin de la misma, punto y coma para separar cada fila ingresada, y un espacio para separar columnas dentro de una fila. Este mismo formato es el que utiliza la aplicación MATLAB que utilizará durante la carrera.
Por ejemplo el ingreso de la siguiente matriz:

$$\begin{bmatrix} 3 & 5 & -1 \\ 0 & 4 & 2 \\ -1 & -7 & 0 \end{bmatrix}$$

Se deberá realizar de la siguiente manera.
[3 5 -1;0 4 2;-1 -7 0]

Para el manejo de las matrices construya una biblioteca de funciones. La misma manejará las matrices a través de una estructura que contenga la cantidad de filas y columnas, y un puntero al espacio de memoria donde se guardará el contenido de la matriz. Por ejemplo:

```
struct Matriz{
        int nFilas;
        int nColumnas;
        float* contenido;
}
```

Para la carga de la matriz implemente una función que procese el string ingresado, obtenga la cantidad de filas y columnas, pida la memoria para el almacenamiento de la misma, cargue cada uno de sus elementos y devuelvo un puntero a la misma.

```
Void ProcesaStringMatriz(char*, struct Matriz*)
```

Construya una segunda biblioteca de funciones que realice las mismas operaciones pero donde la matriz se maneje con un array bidimensional:

```
struct Matriz{
        int nFilas;
        int nColumnas;
        float** contenido;
}
```

La aplicación debe poder ser compilada con cualquiera de las dos bibliotecas indistintamente. Tenga en cuenta que no podrá compilar y linkear el ejecutable con las dos bibliotecas de forma simultánea.

Trabajo Práctico Nº 6

Modifique la aplicación desarrollada a lo largo de los trabajos prácticos anteriores para que permita la evaluación y el cálculo de derivadas e integrales de funciones ingresadas por el usuario a través de una línea de texto. Para ello utilice el solucionador de cálculos del Trabajo Práctico Nº 3. Tenga en cuenta para obtener el valor de la función deberá previamente reemplazar el valor de x por el valor numérico deseado. A su vez deberá definir nuevas funciones de evaluación, derivada e integral para que trabajen con un string en vez de con un puntero a función, ya que en este caso las funciones se definen en tiempo de ejecución.

```
void EvaluaFuncion(float* x, char* funcion, float* y);
void DerivadaFuncion(float* x, char* funcion, float* y);
void IntegralFuncion(float* x, char* funcion, float* y);
```

Adicionalmente la aplicación permitirá el ingreso de múltiples funciones a través de un archivo de texto, a razón de una función por línea del archivo e indicando los valores de xmin, xmax y deltax a utilizar. El formato de cada línea será "%s\t%f,%f, %f\n". Por ejemplo, un archivo con dos funciones a evaluar debería tener el siguiente contenido:

```
X+5/X    1,20,0.1
2+3*X^2 0,5,0.2
```

Este modo de ejecución se podrá indicar directamente por línea de comandos con la opción "-funciones" seguida del path completo de un archivo. Por ejemplo:

```
./Calculador –funciones ./mis_funciones.txt
```

Los resultados de la evaluación de cada función, de su derivada y de su integral se guardarán en un archivo separado por comas, donde cada línea tendrá los valores de x, de la función, de su derivada y su integral. El nombre del archivo estará formado por el nombre de la función y la fecha y hora de ejecución. Ejemplo del contenido de un archivo de salida:

```
x,f(x),derivada,integral
0,1,1,1
1,1.5,0.5,2.5
2,3,1.5,4.5
….
```

Trabajo Práctico Nº 7

Divida la aplicación en un cliente y un servidor concurrente que se comunican a través de un socket.

El cliente manejará un menú de opciones igual que la aplicación realizada hasta el momento, pero enviará todos los datos al servidor que se encargará de realizar las operaciones matemáticas. El cliente recibirá a través de la línea de comando la dirección IP y el puerto del servidor.

Este último generará un proceso hijo por cada pedido, que devolverá el resultado en un formato ascii y luego finalizará su ejecución.

Ejemplo de posible trama enviada del cliente al servidor:

'I'	Menu Nivel 1	Menu Nivel 2	Menu Nivel 3	Menu Nivel 4	MSB Largo Datos	LSB Largo Datos	Dato 0
…..	Dato N-1	'F'					

Ejemplo de posible respuesta del servidor cuando los datos fueron validos:

'I'	MSB Largo Datos	LSB Largo Datos	Dato 0	….	Dato N-1	'F'

Ejemplo de posible respuesta del servidor cuando los datos enviados no fueron validos:

'E'

Apéndice I

Recomendaciones para la Utilzación de SVN en un Proyecto

Si bien el sistema de control de versiones Subversion no impone una estructura de carpetas específica, sino que queda totalmente a la arbitrariedad del usuario, existe una convención para manejar las versiones de un proyecto. En la misma cada proyecto tiene las siguientes carpetas en su raíz:

- trunk: carpeta principal donde trabajaremos en nuestro proyecto. Acá se irá desarrollando el cuerpo principal del mismo.
- tags: carpeta que tiene versiones específicas que deseamos tener disponibles, por ejemplo una versión estable (release) de un programa.
- branches: carpeta en la que tengo una ramificación del proyecto principal. Por ejemplo, si en un momento quiero empezar a evaluar una alternativa a un programa, creo una copia desde el trunk y luego trabajo en el branch. Dependiendo el caso, puedo reintegrar los cambios del branch al trunk. Este uso es avanzado y en esta asignatura no será necesario.

Si se trabaja en un repositorio donde uno podrá manejar varios proyectos o aplicaciones, es recomendado crear una carpeta para cada proyecto en particular. Supongamos que se quiere iniciar a trabajar en este Trabajo Práctico, lo primero que hay que hacer es obtener una copia local de la carpeta de cada alumno en el repositorio:
svn checkout http://svn.electron.frba.utn.edu.ar/info1/r1043/belzunce miRepoInfo1

De esta forma la carpeta miRepoInfo1 va a ser la copia local del repositorio. Si se ha trabajado anteriormente, habrá archivos y carpetas preexistentes. Por ejemplo:
clase1
clase2

En este directorio se debe crear la carpeta para la realización del TP:
mkdir TrabajoPractico

Luego se ingresará a la carpeta del Trabajo Práctico y en su interior se va a generar la estructura recomendada. Para ellos se crean las tres carpetas y se suben al repositorio:
svn add trunk
svn add tags
svn add branches
svn commit -m “Creo la estructura.”

A continuación se puede empezar a trabajar con el primer código fuente:
cd trunk
gedit calculadora.c
svn add callculadora.c
svn comit -m “Agrego la primera versión del código.”
Se continuará trabajando en dicho directorio hasta lograr una versión entregable, por ejemplo para la entrega del TP1. Con dicha versión lista se debe realizar la copia al tag. Para ello, siempre trabajando sobre el trunk, se debe hacer un commit de forma de asegurarse que todos los cambios de la copia local estén confirmados en la versión HEAD del repositorio.
svn commit -m “Versión lista para entregar.”

Con el commit hecho, se puede continuar con la copia al tag. Todas las copias a tags o branches se deben realizar sobre la url y no en la copia local. Subversion me permite realizarlo a nivel de copia

local, pero no es lo más conveniente porque lo que se busca es tener una copia de una versión específica del repositorio. Entonces lo que se hace es:
svn copy http://svn.electron.frba.utn.edu.ar/info1/r1043/belzunce/TrabajosPracticos/trunk svn copy http://svn.electron.frba.utn.edu.ar/info1/r1043/belzunce/TrabajosPracticos/tags/TP1

Por ejemplo, en este caso se copiaría el trunk de la versión X del repositorio en la carpeta TP1 dentro de tags. Habitualmente las carpetas en el tags no se modifican, aunque subversion lo permite (recordamos que subversion simplemente versiona archivos y el usuario es el que elige como manejar el arbol de archivos).
Luego se seguirá trabajando de la misma manera. Si quisiera cambiar la estructura del proyecto siempre es conveniente hacerlo realizando copias de archivos con svn y no agregar y quitarlos manualmente, ya que el historial de las copias también es registrado por subversion.