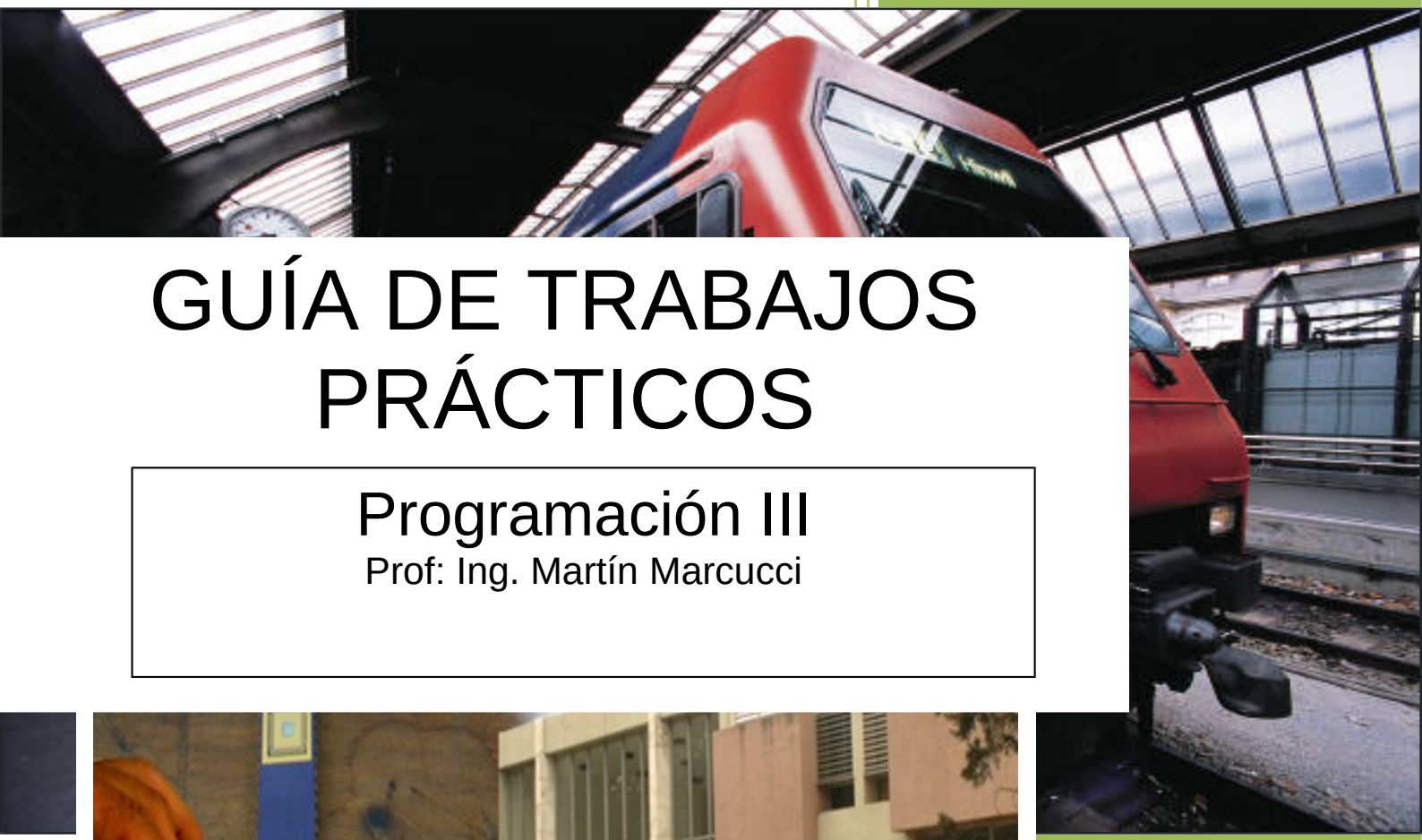
Universidad Católica de Córdoba
Universidad Jesuita

FACULTAD DE INGENIERÍA

ING. EN SISTEMAS
ING. EN COMPUTACIÓN



GUÍA DE TRABAJOS PRÁCTICOS

Programación III

Prof: Ing. Martín Marcucci

Trabajos Prácticos de Aula

ÍNDICE

1. ENCUADRAMIENTO DE LA FORMACIÓN PRÁCTICA

La formación práctica que se desarrolla en esta GTP incluye la resolución de problemas tipo o rutinarios y de problemas abiertos de Ingeniería.

2. CONSIGNAS GENERALES

- *Los problemas propuestos en el presente TP pueden ser resueltos en forma individual o grupal.*
- *De acuerdo a la complejidad creciente que esta asignatura presenta, el planteo o formulación de los problemas debe ser realizado basándose en lo desarrollado en las clases teóricas referidas a la unidad correspondiente y a todas las unidades anteriores ya desarrolladas, como también de aquellas asignaturas que son correlativas de esta.*
- *Es fundamental para un correcto aprendizaje que el alumno controle sus soluciones con la herramienta de desarrollo elegida (Dev-C++, CLion, Code:Blocks, etc).*
- *Se recomienda realizar un repaso de la sintaxis y estructura del lenguaje de programación C++.*

3. INFORME FINAL DEL TRABAJO PRÁCTICO

3.1 Informe de cada Trabajo Práctico

Los alumnos deben presentar por escrito el desarrollo de cada uno de los problemas de esta guía en un informe. Este informe debe contener los siguientes ítems:

1. Objetivo de la práctica
2. Desarrollo de la práctica
3. Conclusiones

3.2 Presentación de la carpeta de trabajos prácticos

Al finalizar el cursado de la asignatura el alumno debe presentar la carpeta de los trabajos prácticos desarrollados. La misma debe contener los siguientes ítems:

1. Carátula, ver: Carátula Presentación de TP.
2. Índice con el detalle de los trabajos prácticos.
3. Los trabajos prácticos.

TRABAJO PRÁCTICO Nº 0

Repaso

A. Objetivos de Aprendizaje

- Recordar estructuras de la programación en C.
- Repasar uso de sintaxis y recursos para aplicar en Estructuras de datos.

B. Consignas a desarrollar en el trabajo práctico

Ejercicio N°1.
Repaso Programación Orientada a Objetos. Implemente una clase abstracta Geometría y 3 clases hijos; Cuadrado, Triangulo y Circulo. Las 3 deben heredar de la clase Geometría. Cree la clase Color y por contención utilícela en las otras tres.

Geometría (abstracta)	Cuadrado	Triangulo	Circulo	Color
Propiedades: • Alto • Ancho • Color Métodos: • getSuperficie • getPerimetro	Propiedades: • Métodos: • getDiagonal	Propiedades: • Angulo Métodos: • getAngulo	Propiedades: • Radio Métodos: • getRadio	Propiedades: • Rojo • Verde • Azul Métodos: • getColor • tenirColor

Ejercicio N°2.
Repaso de uso de Templates en C++. Cree una clase calculadora mediante el uso de Templates, dicha clase debe tener los métodos sumar, restar, dividir, multiplicar que realizaran operaciones en dos propiedades A y B.

Ejercicio N°3.
Repaso de uso de excepciones en C++. En el método dividir de la clase del ejercicio 2, verifique que el divisor no es 0. Si el divisor es 0, arroje una excepción.

Ejercicio N°4.
Calcular el tiempo equivalente en horas minutos y segundos a un número de segundos leído. El resultado debe imprimirse en un formato como el siguiente:
7322 segundos equivalen a: 2 horas, 2 minutos y 2 segundos.

Ejercicio N°5.
En aplicaciones criticas, se usa la redundancia de memoria y procesadores para detectar errores. Implemente la clase Entero con redundancia de valores Int y sobrecargue los operadores +,-,*,/,= para que se comporte como un numero Int común pero arroje excepciones si hay error en la redundancia.

Ejercicio N°6.
Implemente un programa que mida el tiempo que demora un bucle FOR en llenar un array de 1024 elementos int con valores del 0 al 1023.

TRABAJO PRÁCTICO Nº 1
Recursividad

C. Objetivos de Aprendizaje

- Construir los programas desarrollando las funciones recursivas.
- Distinguir la aplicación de la recursividad en problemas reales.

D. Unidad temática que incluye este trabajo práctico

Este trabajo práctico corresponde a la unidad Nº: 1 de la programación de la asignatura.

E. Consignas a desarrollar en el trabajo práctico

Ejercicio Nº1.

Implemente una función para potencias enteras con recursividad.

Definición recursiva para elevar un número a una potencia: Un número elevado a la potencia cero produce la unidad; la potencia de un número se obtiene multiplicándolo por sí mismo elevando a la potencia menos uno.

Por ejemplo: $3^2=3\cdot(3^1)=3\cdot(3\cdot 3^0)=3\cdot(3\cdot 1)=3\cdot 3=9$

Ejercicio Nº2.

Implemente una función recursiva que, teniendo un array ingresado por teclado, me devuelva la suma de todos sus elementos

Ejercicio Nº3.

Algoritmo de Ackerman -> Número de Combinaciones

Recursivamente, podemos definir el número de combinaciones de m objetos tomados de n, denotado:

$$A(m,n)=\begin{cases} n+1, & \text{si m=0;} \\ A(m-1,1), & \text{si m > 0 y n=0;} \\ A(m-1,A(m,n-1)), & \text{si m > 0 y n > 0;} \end{cases}$$

Ejercicio Nº4,

Algoritmo de Euclides: mostrar que el máximo común divisor (mcd) de **a** y **b**, (**a** > **b** > 0), es igual a **a** si **b** es cero, en otro caso es igual al mcd de **b** y el remanente de **a** dividido por **b**, si **b** > 0.

Ejercicio Nº5.

Escribir segmentos de programa que lleven a cabo de forma recursiva, cada una de las siguientes tareas:

a) Calcule la parte entera del cociente, cuando el entero a se divide por el entero b.

b) Calcule el resto entero, cuando el entero a es dividido por el entero b.

c) Utilice los módulos a) y b) para escribir una función recursiva que dado un entero no negativo lo imprima como una serie de dígitos separados por espacios.

Ej: dado 1024, debe escribir 1 0 2 4

Ejercicios complementarios de repaso

Ejercicio R1:
Es interesante ver en acción a la recursión. Modifique las funciones echas en los ejercicios anteriores para que la función recursiva muestre por pantalla en cada linea, los valores con los que fue llamada.

Ejercicio R2:
Escriba funciones recursivas que realicen las siguientes operaciones matemáticas:

- Producto mediante sumas sucesivas
- Cociente mediante restas sucesivas
- Factorial de un número
- Raíz cuadrada de un numero mediante algoritmo babilónico

Ejercicio R3:
Escriba funciones recursivas, que dado un numero decimal lo muestren en pantalla en notación hexadecimal, octal y binaria .

TRABAJO PRÁCTICO Nº 2
Listas

A. Objetivos de Aprendizaje

- Comprender la estructura de una lista dinámica en C++
- Construir programas utilizando listas dinámicas en C++
- Distinguir el uso de Listas en casos reales

B. Unidad temática que incluye este trabajo práctico

Este trabajo práctico corresponde a la unidad Nº:2 de la programación de la asignatura.

C. Consignas a desarrollar en el trabajo práctico

Ejercicio N°1.
Implementar la función **fnInvierte(lista)**. Esta función invertirá el orden original de los elementos en la lista, de tal forma que el último elemento será ahora el primero, el penúltimo será el segundo, y así hasta que el primero sea el último. Considere que la lista no está vacía y que no se construirá una nueva, sólo se invertirá el orden de los elementos de la lista original.

Ejercicio N°2.
Escribir un programa que permita agregar nº enteros a una lista de números aceptada por teclado. El programa pregunta si debe agregar al principio, al final o en el medio y agrega el elemento a la lista.

Ejercicio N°3.
Escribir un programa que permita eliminar elementos de una lista de números aceptada por teclado. El programa pregunta qué elemento borrar considerando que el primero por la izquierda es el 1.

Ejercicio N°4.
Crear una función que dada dos listas enlazadas, pasadas como parámetro, devuelva una lista enlazada que es la unión de las otras dos.

Ejercicio N°5.
Escriba la clase CircList para implementar una lista circular

Ejercicio N°6.
Agregue el método: **insertAfter2(int oldValue, int n, int newValue)** que inserte un nodo con el valor newValue después de la enésima ocurrencia de oldValue.

TRABAJO PRÁCTICO Nº 3
Pilas

A. Objetivos de Aprendizaje

- Comprender la estructura de una Pila dinámica en C++
- Construir programas utilizando Pilas en C++
- Distinguir el uso de Pilas en casos reales

B. Unidad temática que incluye este trabajo práctico

Este trabajo práctico corresponde a la unidad Nº:3 de la programación de la asignatura.

C. Consignas a desarrollar en el trabajo práctico

Ejercicio N°1.
Escriba un programa que introduzca una palabra y utilice una pila para imprimir la misma palabra invertida.

Ejercicio N°2.
Escriba una función que indique si dos pilas son iguales. Se entiende que dos pilas son iguales cuando sus elementos son idénticos y aparecen en el mismo orden.

Ejercicio N°3.
Agregar la función peek a la clase pila que permita ver el valor del tope de pila sin sacarlo.

Ejercicio N°4. Escriba un programa que ingresada una función matemática, informe si la cantidad de (, [, { que abren están balanceados con los que cierran.

Ejercicio N°5. Se tiene la siguiente expresión infija:
$(6+2)*5-8/4$ Realizar un programa en C++ que me calcule la expresión postfija.

TRABAJO PRÁCTICO Nº 4
Colas

A. Objetivos de Aprendizaje

- Comprender la estructura de una Cola dinámica en C++
- Construir programas utilizando Colas en C++
- Distinguir el uso de Colas en casos reales

B. Unidad temática que incluye este trabajo práctico

Este trabajo práctico corresponde a la unidad Nº: 4 de la programación de la asignatura.

C. Consignas a desarrollar en el trabajo práctico

Ejercicio N°1.
Diseñe un programa que sea capaz de leer dos colas y mediante un mensaje indicar si son iguales. Nota: los elementos constitutivos de las colas son caracteres.

Ejercicio N°2.
Utilizando una pila y una cola, realizar una función que reciba una cadena y devuelva el valor lógico verdadero si dicha cadena es un palidromo. No se deben tener en cuenta los espacios y signos de puntuación.
Ejemplos de palidromo:

anita lava la tina
la ruta natural
la ruta nos aporto otro paso natural

Ejercicio N°3.
Agregue un método a la clase cola para insertar un nodo según un valor entero de prioridad. El valor 0 es la máxima prioridad. Este método debe colocar el nodo lo mas próximo al frente de la cola, pero atrás del ultimo nodo con la misma prioridad.

Ejercicio N°4.
En un supermercado, se tiene sólo una caja habilitada para que los clientes puedan pagar sus compras. La caja tiene prioridad para mujeres embarazadas. Dada esta situación, se pide que se implemente un sistema que calcule la cantidad de productos comprados por cada cliente y el monto total gastado, también la cantidad de embarazadas que accedieron a la cola.

TRABAJO PRÁCTICO Nº 5
Árboles

A. Objetivos de Aprendizaje

- Comprender la estructura de un Árbol dinámico en C++
- Construir programas utilizando Árboles en C++
- Distinguir el uso de Árboles en casos reales

B. Unidad temática que incluye este trabajo práctico

Este trabajo práctico corresponde a la unidad Nº: 5 de la programación de la asignatura.

C. Consignas a desarrollar en el trabajo práctico

Ejercicio N°1.
Diseñar una aplicación que permita:
Cargar en un array 30 números generados aleatoriamente entre 100 y 500.
Imprimir los números.
Crear un árbol binario de búsqueda con los datos del array.

Ejercicio N°2.
Reconstruir un árbol binario a partir de los recorridos siguientes:
Preorden: 2, 5, 3, 9, 7, 1, 6, 4, 8.
Inorden: 9, 3, 7, 5, 1, 2, 6, 8, 4.
Inorden: 5, 6, 12, 10, 1, 9, 13, 4, 8, 2, 7, 3, 11.
Postorden: 6, 5, 10, 9, 1, 13, 12, 2, 8, 3, 11, 7, 4.

Ejercicio N°3.
Implementar el método contarPorNivel que devuelve el número de nódos del nivel iésimo de un árbol binario.

Ejercicio N°4.
Especificar la operación espejo que devuelve la imagen especular de un árbol binario.

TRABAJO PRÁCTICO Nº 6
Tablas de Dispersión - Hash

A. Objetivos de Aprendizaje

- Comprender la estructura de una tabla de Hash en C++
- Construir programas utilizando Tablas de Hash en C++
- Distinguir el uso de Tablas de Hash en casos reales

B. Unidad temática que incluye este trabajo práctico

Este trabajo práctico corresponde a la unidad Nº: 6 de la programación de la asignatura.

C. Consignas a desarrollar en el trabajo práctico

Ejercicio N°1.
Probar el algoritmo visto en clase agregándole una función main.

Ejercicio N°2.
La manera más simple de resolver una colisión es construir, para cada localización de la tabla, una lista enlazada de registros cuyas claves caigan en esa dirección. Este método se conoce normalmente con el nombre de encadenamiento separado y obviamente la cantidad de tiempo requerido para una búsqueda dependerá de la longitud de las listas y de las posiciones relativas de las claves en ellas.
Modificar el algoritmo visto en clase para que acepte listas enlazadas.

Ejercicio N°3.
Modificar el algoritmo visto en clase para que puedan eliminarse los registros. Una vez eliminados, pueda hacerse un rehashing de la tabla.

TRABAJO PRÁCTICO Nº 7
Grafos

A. Objetivos de Aprendizaje

- Comprender la estructura de grafo implementado en C++
- Construir programas utilizando grafos en C++
- Distinguir el uso de grafos en casos reales

B. Unidad temática que incluye este trabajo práctico

Este trabajo práctico corresponde a la unidad Nº: 7 de la programación de la asignatura.

C. Consignas a desarrollar en el trabajo práctico

Ejercicio N°1.
Implemente una clase Grafo utilizando una matriz

Ejercicio N°2.
Implemente una clase Grafo utilizando nodos enlazados

Ejercicio N°3.
Implemente el algoritmo de Dijkstra para conocer la distancia mínima entre un nodo dado y el resto del los del grafo que sean alcanzables.

TRABAJO PRÁCTICO Nº 8
Tablas de Dispersión - Ordenamiento

A. Objetivos de Aprendizaje

- Comprender los algoritmos de ordenamiento
- Construir programas algorítmicos de ordenamiento en C++
- Distinguir el uso de algoritmos de ordenamiento en casos reales

B. Unidad temática que incluye este trabajo práctico

Este trabajo práctico corresponde a la unidad Nº: 8 de la programación de la asignatura.

C. Consignas a desarrollar en el trabajo práctico

NOTA: Utilice como entrada de datos de los algoritmos de ordenamiento el archivo dic.txt. Para cada uno de los siguientes ejercicios indique la cantidad de condiciones realizadas para realizar el ordenamiento

Ejercicio N°1.
Implemente la función **sortBubble** que recibe un vector y lo ordena mediante el método Burbuja.

Ejercicio N°2.
Implemente la función **interSort** que recibe un vector y lo ordena mediante el algoritmo de ordenamiento por intercalación.

Ejercicio N°3.
Implemente la función **quickSort** que recibe un vector y lo ordena mediante el algoritmo de ordenamiento rápido.

Ejercicio N°4.
Implemente la función **shellSort** que recibe un vector y lo ordena mediante el algoritmo de ordenamiento shell.

Ejercicio N°5.
La ordenación de tipo burbuja es ineficiente en caso de arreglos grandes. Haga las siguientes modificaciones en el ej1 para mejorar el rendimiento de este tipo de ordenación:

a) Después de la primera pasada el número más alto está garantizado que deberá aparecer en el elemento numerado más alto del arreglo; después de la segunda pasada, los dos números más altos estarán en su lugar, y así en lo sucesivo.

b) Los datos en el arreglo pudieran ya estar en orden casi apropiado. Modifique la ordenación para verificar si se han hecho intercambios. Si no se han hecho intercambios, entonces los datos ya deberían estar en el orden apropiado y, por lo tanto, el programa debe

darse por terminado. Si ha habido intercambio, entonces por lo menos se requiere una pasada adicional.

TRABAJO PRÁCTICO Nº 9
Tablas de Dispersión - Búsqueda

A. Objetivos de Aprendizaje

- Comprender los algoritmos de Búsqueda
- Construir programas algorítmicos de Búsqueda en C++
- Distinguir el uso de algoritmos de Búsqueda en casos reales

B. Unidad temática que incluye este trabajo práctico

Este trabajo práctico corresponde a la unidad Nº: 8 de la programación de la asignatura.

C. Consignas a desarrollar en el trabajo práctico

Ejercicio N°1.

Describir qué hace la siguiente función; si es recursiva o iterativa. Si es una función recursiva, realizar su versión iterativa. Si es una función iterativa, realizar su versión recursiva.

```
void fn(int x, int v[], int ini, int fin, int pos, bool retorno)
{
        pos = ini;
        retorno =false;
        while ( (pos <=fin) && (!retorno) ){
                if (v[pos]==x){
                        retorno =true;
                }
                else
                        ++pos;
        }
}
```

Ejercicio N°2.

Implementar los métodos “buscar” y “buscarRecursiva” de la clase lista.

Ejercicio N°3.

Escribir un algoritmo que dado una secuencia de números ingresados por teclado vaya generando un árbol balanceado. Modificar la Clase Árbol vista en clase.