DIIT
Departamento de Ingeniería e Investigaciones Tecnológicas

TÓPICOS DE PROGRAMACIÓN

Universidad Nacional de La Matanza

Guía de Trabajos Prácticos

Año 2023

NOTA: el número de ejercicio entre paréntesis corresponde al que originalmente posee en la guía de TP de Programación 1110

Ejercicio x.y Ejercicios nuevos.

Ejercicios de repaso:

Ejercicio 1

El factorial de un número natural incluido el 0, se calcula de la siguiente manera:

$$N! = \begin{cases} 1 & \text{si } N = 0 \\ N.\,(N-1)! & \text{si } N > 0 \end{cases}$$

o sea, N! = N. (N - 1). (N - 2). …. 3. 2. 1

Ejemplo: 5! = 5. 4. 3. 2. 1 = 120

Desarrollar una función para calcular el factorial de un entero.

Ejercicio 2

Dados dos números enteros m y n (m >= n y n >= 0), el número combinatorio se calcula de la siguiente manera:

$$\binom{m}{n} = \frac{m!}{n!\,(m-n)!}$$

Desarrollar una función para calcular el combinatorio m sobre n.

Ejercicio 3

Dado un número entero X y una tolerancia (TOL), puede obtenerse ex mediante la suma de los términos de la serie:

$$e^x = 1 + \left(\frac{x^1}{1}\right) + \left(\frac{x^2}{1\cdot 2}\right) + \left(\frac{x^3}{1\cdot 2\cdot 3}\right) + \left(\frac{x^4}{1\cdot 2\cdot 3\cdot 4}\right) + \ldots$$

El proceso termina cuando se obtiene un término calculado que sea menor que la tolerancia TOL.

Desarrollar una función para calcular el ex , dados X y TOL.

Ejercicio 4

La raíz cuadrada de un número positivo A puede calcularse mediante un proceso iterativo que genera términos según la siguiente fórmula:

$$R_1 = 1$$

$$R_i = \frac{1}{2}\cdot\left(R_{i-1} + \left(\frac{A}{R_{i-1}}\right)\right)$$

El proceso de cálculo se da por terminado cuando la diferencia entre dos términos sucesivos es menor que una cota fijada de antemano.

Desarrollar una función para calcular la raíz cuadrada de X con una tolerancia TOL.

Ejercicio 5

En la serie de Fibonacci, cada término es la suma de los dos anteriores y los dos primeros términos son 1

Serie: 1 1 2 3 5 8 13 21 34 …

Desarrollar una función para determinar si un entero pertenece a la serie de Fibonacci.

Ejercicio 6

Dados X y una tolerancia TOL es posible calcular el seno (x) mediante la suma de los términos de la serie:

$$seno(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \dots$$

Este proceso continúa mientras el término calculado (en valor absoluto) sea mayor que la tolerancia. Desarrollar una función que obtenga el seno de X con tolerancia TOL, utilizando dicha serie.

Ejercicio 7

Un número natural es perfecto, deficiente o abundante según que la suma de sus divisores positivos menores que él sea igual, menor o mayor que él. Por ejemplo:

Número	Divisores positivos menores que él	Suma de los divisores	Clasificación
6	1, 2, 3	6	PERFECTO
10	1, 2, 5	8	DEFICIENTE
12	1, 2, 3, 4, 6	16	ABUNDANTE

Desarrollar una función que determine si un número natural es perfecto, deficiente o abundante.

Ejercicio 8

Dados dos números naturales (incluido el cero), obtener su producto por sumas sucesivas.

Ejercicio 9

Dados dos números naturales A y B, desarrollar una función para obtener el cociente entero A/B y el resto. (A puede ser 0; B, no).

Ejercicio 10

Construir un programa que lea un número natural N y calcule la suma de los primeros N números naturales.

Ejercicio 11

Construir un programa que lea un número natural N y calcule la suma de los primeros N números pares.

Ejercicio 12

Construir un programa que lea un número natural N y calcule la suma de los números pares menores que N.

Ejercicio 13

Desarrollar una función que determine si un número natural es primo.

Unidad 1: Arrays

Arrays unidimensionales. Concepto de puntero. Aritmética de Punteros. Manejo de arrays unidimensionales con diferentes notaciones. Arrays numéricos. Cadenas. Funciones de biblioteca. Arrays multidimensionales. Manejo de arrays multidimensionales con diferentes notaciones. Uso de arrays.

Ejercicios con vectores (Arreglos unidimensionales).

Ejercicio 1.1 (TP1 ejercicio 22)

Desarrollar una función que inserte un elemento en un arreglo de enteros, dada la posición de inserción.

Ejercicio 1.2 (TP1 ejercicio 23)

Desarrollar una función que inserte un elemento en un arreglo de enteros, ordenado en forma ascendente, de forma de no alterar el orden.

Ejercicio 1.3 (TP1 ejercicio 24)

Desarrollar una función que elimine el elemento que ocupa una determinada posición de un arreglo de enteros.

Ejercicio 1.4 (TP1 ejercicio 25)

Desarrollar una función que elimine la primera aparición de un elemento determinado de un arreglo de enteros.

Ejercicio 1.5 (TP1 ejercicio 26)

Desarrollar una función que elimine todas las apariciones de un determinado elemento de un arreglo de enteros.

Ejercicio 1.6 (TP1 ejercicio 27)

Desarrollar una función que determine si una cadena de caracteres es un palíndromo.

Ejercicio 1.7 (TP1 ejercicio 28)

Desarrollar una función que devuelva el valor numérico de una cadena de caracteres (asumiendo que los caracteres representan dígitos).

Ejercicio 1.8

Desarrollar una función que cuente el número de apariciones de una palabra dentro de una cadena de texto. Para ello la función recibe como parámetros dos punteros a char. El primero corresponde al texto, el segundo corresponde a la cadena buscada. La función debe retornar el número de ocurrencias. Contemple las condiciones de borde y haga un listado de éstas.

Ejercicio 1.9

Desarrollar una función que normalice la cadena de texto que se le pasa como argumento. En este caso, la cadena se encontrará normalizada cuando la primera letra de cada palabra sea mayúscula y las siguientes minúsculas. La cadena normalizada no deberá contener espacios o tabulaciones al inicio o al final. En el caso de que las palabras de la cadena se encuentren separadas por más de un espacio o tabulación, se deberán eliminar los excedentes. Se debe modificar la cadena pasada como argumento. No puede realizar una o más copias locales de la cadena original. Contemple las condiciones de borde y haga un listado de éstas.

Ejercicios con matrices (Arreglos bidimensionales)

La definición adecuada de los parámetros de las siguientes funciones es parte de la ejercitación.

Ejercicio 1.10 (TP1 ejercicio 29)

Desarrollar una función para que, dada una matriz cuadrada de reales de orden N, obtenga la sumatoria de los elementos que están por encima de la diagonal principal (excluida ésta). Lo mismo para la diagonal secundaria. Lo mismo incluyendo la diagonal. Lo mismo, con los que están por debajo de la diagonal.

Ejercicio 1.11 (TP1 ejercicio 30)

Desarrollar una función para que, dada una matriz cuadrada de enteros de orden N, obtenga la traza de la misma (sumatoria de los elementos de la diagonal principal). Lo mismo, pero con la diagonal secundaria.

Ejercicio 1.12 (TP1 ejercicio 31)

Desarrollar una función que determine si una matriz cuadrada de enteros de orden N es matriz diagonal (ceros en todos sus elementos excepto en la diagonal principal).

Ejercicio 1.13 (TP1 ejercicio 32)

Desarrollar una función que determine si una matriz cuadrada de enteros de orden N es matriz identidad (matriz diagonal, con unos en la diagonal principal y ceros en los restantes).

Ejercicio 1.14 (TP1 ejercicio 33)

Desarrollar una función que determine si una matriz cuadrada de enteros de orden N es simétrica.

Ejercicio 1.15 (TP1 ejercicio 34)

Desarrollar una función para trasponer, in situ, una matriz cuadrada.

Ejercicio 1.16 (TP1 ejercicio 35)

Desarrollar una función para obtener la traspuesta de una matriz dada.

Ejercicio 1.17 (TP1 ejercicio 36)
Desarrollar una función para obtener la matriz producto entre dos matrices de enteros.

Ejercicio 1.18 (TP1 ejercicio 37)
Se dispone de una matriz cuadrada de enteros de orden **N**, donde cada elemento **[i][j]** representa la cantidad de puntos que obtuvo el equipo **i** frente al equipo **j** al fin de un torneo de fútbol (partidos de ida y vuelta) en el que participaron **N** equipos. El sistema de puntuación es: 3 puntos para el ganador del partido y ninguno para el perdedor; 1 punto para cada equipo en caso de empate.
Desarrollar una función que determine si la matriz está correctamente generada.
Desarrollar una función que genere un arreglo de **N** elementos tal que cada elemento **V[k]** contenga la cantidad de puntos obtenidos por el equipo **k**.

Ejercicios ampliados con vectores (Arreglos unidimensionales)

Ejercicio 1.19
Haga una reingeniería de los ejercicios 1.1 a 1.5 para que puedan operar con cualquier tipo de dato (recuerde que los ejercicios originales estaban diseñados para operar con números enteros). Haga uso de memoria estática.

Unidad 2: Tipos de Memorias

Tipos de memoria. Memoria de pila. Memoria Heap. Manejo de memoria en tiempo de ejecución. Ventajas y desventajas de cada tipo de memoria.

Ejercicio 2.1
Desarrolle un programa que reserve memoria para almacenar 10 enteros. Luego asigne valores a dichos elementos. Asegúrese de devolver la memoria reservada.

Ejercicio 2.2
Modifique el programa del punto 2.1 para que la cantidad de elementos sea ingresada por teclado.

Ejercicio 2.3
Desarrolle un programa que reserve memoria para almacenar 5 elementos de un tipo de dato creado por Usted. Dicho tipo de dato debe contener al menos un array de char, un entero, un flotante y un char. Luego asigne valores a dichos elementos. Asegúrese de devolver la memoria reservada.

Ejercicio 2.4
Implemente utilizando memoria dinámica las siguientes funciones:

a) Una función que retorna una copia de la cadena pasada como parámetro con exactamente la misma cantidad de caracteres de la cadena origen. Ej:

```
char* copiaCadena(const char* origen)
```

b) Una función que genere una copia de un elemento pasado como parámetro, este puede ser un tipo primitivo (int,float,char) una estructura o incluso un vector. Ej:

```
void* copiaCosas(void* elemento, unsigned tam)
```

Estas funciones tienen un objetivo estrictamente didáctico, y como puede observar persiguen objetivos similares a las funciones de biblioteca de C pero su implementación es totalmente diferente. ¿Por qué no se recomienda este tipo de implementación en sistemas productivos?

Ejercicio 2.5

Implemente su propia versión de la función de biblioteca memmove, asegurando que los datos se mueven correctamente incluso si hay superposición entre ellos. Verifique que no tiene pérdidas de memoria (memory leaks) .

Unidad 3: Pasaje de Parámetros

Concepto de pasaje de parámetros. Pasaje de parámetros por copia o valor. Pasaje de parámetros por dirección. Punteros a variables. Punteros a funciones. Funciones de biblioteca de ordenamiento y búsqueda. Creación de funciones independientes del tipo de dato. Concepto de funciones map, filter y reduce. Argumentos en el main.

Ejercicio 3.1 (TP2 ejercicio 1)

Escriba una función que permita desplegar un menú de opciones, devolviendo una opción válida.

Escriba una función que reciba por argumento la dirección de comienzo de un array de float y la cantidad máxima de elementos a ingresar (no utilice subíndices). La función permitirá terminar el ingreso con una condición fijada por el alumno y devolverá la cantidad de elementos ingresados (puede ser cero).

Escriba una función que permita buscar el mínimo elemento de un array de float.

Escriba una función que determine el promedio de los elementos que se encuentran en las posiciones pares de un array de float.

Escriba una función que muestre los elementos de un array de float en orden inverso.

Escriba una función que almacene en un archivo de texto los elementos de un array de float, a razón de un flotante por línea de texto.

Haciendo uso de las funciones anteriores, escriba un programa que al comenzar su ejecución permita el ingreso para un array de float, luego de lo cual muestre un menú de opciones para:

1- Buscar el mínimo elemento,
2- Calcular el promedio de los valores de las posiciones pares,
3- Mostrarlo en orden inverso,
4- Salir.

Consulte de qué modo puede hacer que el programa trabaje con otros tipos de datos (double, long double, int, unsigned, etc.), con mínimas modificaciones.

Ejercicio 3.2 (TP2 ejercicio 2)
Escriba una función que devuelva en qué dirección de memoria se encuentra un elemento dentro de un array. Si el elemento no se encuentra, debe devolver NULL.

Ejercicio 3.3 (TP2 ejercicio 3)
Escriba una función que permita el ingreso de una cantidad variable de elementos en un array de enteros int.
Escriba una función que calcule la suma de todos los elementos almacenados en un array de enteros, y su promedio. El promedio (float) debe ser devuelto por la función, y la suma debe ser también devuelta mediante un argumento extra (puntero a long) que recibe la función.
Escriba otra versión de la función anterior, pero devolviendo ambos valores calculados en una variable que responda a una estructura compuesta de un miembro long y un miembro float.
a- escriba un main que utilice la primera y segunda función, y
b- otro main que utilice la primera y la tercera
En ambos casos, la suma y el promedio deben ser mostrados en la función main.
¿Tiene claro que la primera alternativa es mejor que la segunda porque no es necesario el uso de una estructura? En ciertos casos puede ser mejor la segunda alternativa.

Ejercicio 3.4 (TP2 ejercicio 4)
Existen funciones de conversión (atoi, itoa, atol, etc., declaradas en la biblioteca stdlib.h), que usted debería conocer y recordar. Escriba, compile y ejecute un programa en que haga uso de tales funciones de conversión.

Ejercicio 3.5 (TP2 ejercicio 5)
En la biblioteca stdio.h hay dos funciones que permiten obtener idénticos (o similares) resultados, se trata de sscanf y sprintf. Escriba, compile y ejecute un programa en que utilice estas funciones.

Ejercicio 3.6 (TP2 ejercicio 6)
Escriba una macro que:

a) redondee un número real al entero más cercano.
b) calcule el valor absoluto de un argumento
c) retorne la parte entera de un número
d) retorne la parte decimal
e) verifique si el número es par
f) reciba dos argumentos (x,y) verifique si el número “x” es potencia del número “y”
g) diga si el argumento es letra
h) es dígito
i) es mayúscula
j) es minúscula
k) es blanco
l) convierta un caracter a mayúscula
m) convierta un caracter a minúscula

Escriba una función que devuelva el menor entre dos enteros que recibe por argumento.
Escriba una macro que cumpla con el mismo cometido.

Ejercicio 3.7

Investigue y utilice las macros max y min de la biblioteca stdlib.h.

Ejercicio 3.8

Escriba una función que intercambie dos enteros que recibe por puntero.
Escriba una macro multilínea que cumpla con el mismo cometido.

Ejercicio 3.9

Dado un arreglo enteros, utilice la función qsort (biblioteca <stdlib>) para ordenarlos de menor a mayor. Repita el ejercicio ordenándolos de mayor a menor.
Para los siguientes ejercicios de ordenamiento o búsqueda, asuma la existencia de:

```
typedef struct
{
      int dni;
      char apellido[20];
      char nombres[30];
      float peso;

} tPersona;
```

Ejercicio 3.10

Dado un arreglo de elementos de tipo tPersona, utilice la función qsort (biblioteca <stdlib>) para ordenarlos por DNI.

Ejercicio 3.11

Dado el arreglo de elementos de tipo tPersona del ejercicio 3.10, utilice la función qsort (biblioteca <stdlib>) para ordenarlos por APELLIDO y NOMBRE.

Ejercicio 3.12

Dado el arreglo de elementos de tipo tPersona del ejercicio 3.10, diseñe e implemente la función:

```
      int buscarXdni(const tPersona *p, tPersona *d);
```

El propósito de la función es realizar una búsqueda por DNI de una persona dentro de un arreglo de elementos del tipo persona. En caso de encontrarse el DNI contenido en el campo dni del segundo argumento, se deben completar los demás campos con los valores correspondientes al elemento localizado en el arreglo y además la función debe retornar 1. En el caso de que la búsqueda no haya sido exitosa se debe retornar 0.

Ejercicio 3.13

Idem ejercicio 3.12 buscando por APELLIDO y NOMBRE. Adecúe el nombre de la función respetando la firma y el tipo de valor devuelto.

Ejercicio 3.14

Diseñe e implemente un programa que reciba tres valores desde la línea de comando. Los primeros dos valores deberán ser números enteros. El tercer valor será el tipo de operación (+: sumar, -: restar, *: multiplicar, /: dividir). El programa deberá imprimir el resultado de la operación. Establezca las condiciones de borde. En caso de que se produzcan errores deben informarse por pantalla.

Ejercicio 3.15

Desarrolle una función de intercambio genérica tal que pueda intercambiar 2 bloques de información de manera independiente al tipo de dato. No utilice memoria dinámica.

Ejercicio 3.16

Desarrolle una función genérica que encuentre el menor elemento dentro de un vector. Verifique su funcionamiento encontrando el menor entero, el menor flotante, y el menor alumno (Por DNI, nombre y apellido o promedio). La función retornará la dirección del elemento menor

Ejercicio 3.17

Desarrolle una función genérica que permita insertar de forma ordenada un elemento en un vector. Puede ayudarse, usando como patrón, la versión no genérica desarrollada y probada de la práctica 1 y con las funciones de biblioteca memcpy y memmove.

Ejercicio 3.18

Ordenamiento genérico - Desarrolle una función genérica que ordene un vector. Puede ayudarse con la función intercambio y buscar menor desarrolladas en ejercicios anteriores e implementar un algoritmo de selección.

Unidad 4: Tipo de Dato Abstracto (TDA)

Concepto de Tipo de Dato Abstracto. Creación e implementación de un TDA. Diferencia entre TDA y estructuras de datos.

Para los siguientes ejercicios con fechas, asuma la existencia de:

```
typedef struct
{
      int dia;
      int mes;
      int anio;
} tFecha;
```

Ejercicio 4.1 (TP1 ejercicio 14)
Desarrollar una función que determine si una fecha es formalmente correcta.

Ejercicio 4.2 (TP1 ejercicio 15)
Desarrollar una función que a partir de una fecha obtenga la correspondiente al día siguiente.

Ejercicio 4.3 (TP1 ejercicio 16)
Desarrollar una función que a partir de una fecha obtenga la que resulte de sumarle N días.

Ejercicio 4.4 (TP1 ejercicio 17)
Desarrollar una función que a partir de una fecha obtenga la que resulte de restarle N días.

Ejercicio 4.5 (TP1 ejercicio 18)
Desarrollar una función que a partir de dos fechas obtenga la cantidad de días que hay entre ellas.

Ejercicio 4.6 (TP1 ejercicio 19)
Desarrollar una función que a partir de una fecha devuelva un entero que representa el día de la semana que le corresponde (0: Domingo; 1: lunes; 2:Martes;…etc.)

Ejercicio 4.7
Implemente un TDA Vector. Debe desarrollar una versión con memoria estática y otra con memoria dinámica. Debe implementar las primitivas:

- crear vector
- vector lleno
- vector vacío
- insertar elemento en orden
- eliminar elemento
- destruir vector

Tenga en cuenta que la primitiva destruir debe según sea el caso liberar la memoria reservada, o poner la cantidad de elementos en cero.

Unidad 5: Persistencia de memoria

Archivos binarios y de texto, su creación, modos de acceso, posicionamiento, cierre, eliminación, y funciones relacionadas. Creación de archivos como lotes de prueba. Merge o apareo de archivos.

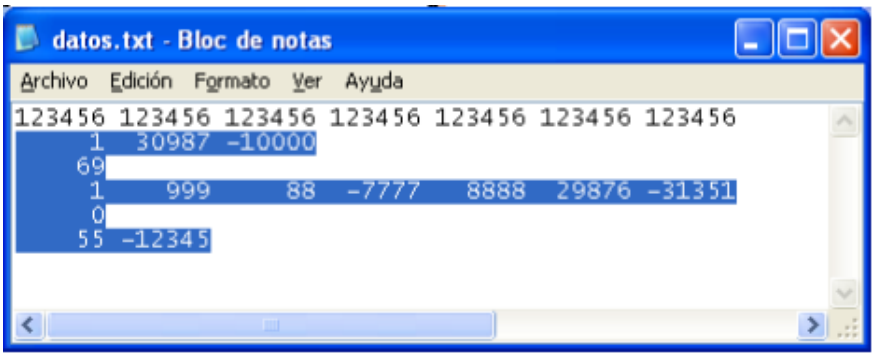
Ejercicio 5.1 (TP2 ejercicio 7)
Escriba una función booleana que permita abrir un archivo, mostrando o no un mensaje de error por stdout, según el valor de un argumento.
Escriba una macro multilínea que cumpla con el mismo cometido.

Ejercicio 5.2 (TP2 ejercicio 8)
Dado un array de char que contiene un texto compuesto por palabras que termina en '.' (o en su defecto en carácter nulo -'\0'-), escriba un programa en que determine e informe:

a- cuál es la primera palabra y cuántas veces se repite en el texto
b- cuántas palabras tiene el texto
c- longitud de la palabra más larga

El siguiente es un ejercicio compuesto por los ejercicios 5.3, 5.4 y 5.5. Debería resolverlo en forma progresiva.

Ejercicio 5.3 (TP2 ejercicio 9)
Escriba un programa que genere un archivo de texto ("datos.txt") a partir del ingreso por teclado de números enteros, de modo que en cada línea de texto haya una cantidad variable de cadenas de caracteres que representen tales números. En el archivo debe haber como mínimo la representación de un entero y como máximo de siete. La separación entre estas cadenas que representan números debe ser de al menos un carácter espacio (' ') o a lo sumo cinco, de modo que queden alineados por la derecha al leer el contenido del archivo con un procesador de texto como el 'Notepad' o 'Bloc de Notas'.
Ingrese los enteros con una variable short int (note que el rango de las mismas pertenece al intervalo [-32768, 32767]). Vea e interprete qué sucede cuando ingresa números fuera de ese rango.



Note que:

- los caracteres de separación sólo están entre números.
- la primera línea se ha indicado para visualizar la alineación de los números.
- en el archivo debe haber al menos una línea con un número y otra con siete.

Utilice el generador de números pseudo aleatorios para determinar cuántos números se almacenan por línea de texto, y además para que determine cuántas líneas se almacenarán una vez cumplida la condición de una línea con un número y otra con siete.

Ejercicio 5.4 (TP2 ejercicio 10)
Escriba una función que determine si una cadena de caracteres que representa a un número, es decir compuesta por los caracteres que representan dígitos, recibida por argumento:

- es capicúa (es obvio),
- es múltiplo de 5 (el último dígito es 0 ó 5),
- es múltiplo de 6 (es par -termina en '0', '2', '4', '6', '8'; y la suma de sus dígitos es múltiplo de 3),
- es múltiplo de 11 (la suma de los dígitos de posiciones pares, y la suma de los dígitos de posiciones impares es múltiplo de 11),
- es mayor que '100' (o cualquier otra cadena representando un número entero cualquiera, p. ej.: '-42').

Escriba una función que valide si todos los caracteres de una cadena representan un número dentro del intervalo de los: short int.

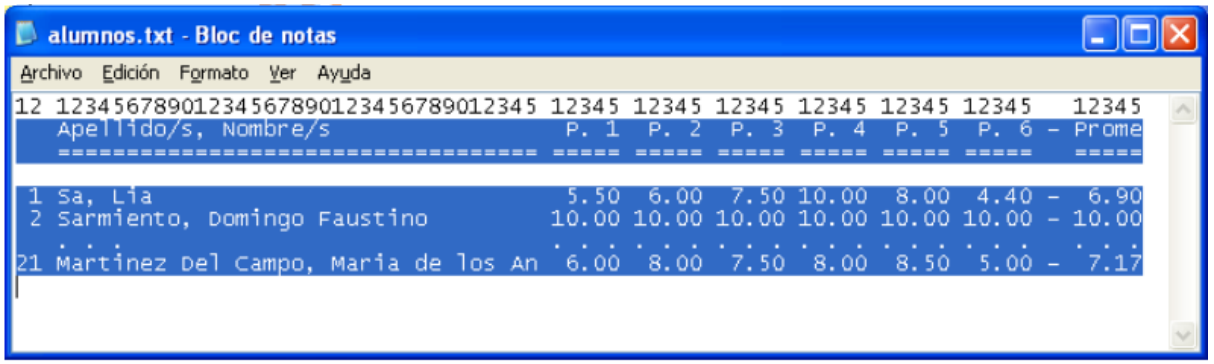
Ejercicio 5.5 (TP2 ejercicio 11)
Leyendo (sólo una vez) un archivo de texto como el del <ej.: 9> y utilizando las funciones del <ej.: 10>, y otras al efecto, determinar:

- cuántos son múltiplo de 5,
- cuántos son múltiplo de 6,
- cuántos son múltiplo de 11, y
- generar un archivo con los que sean mayores que '100' (o cualquier otro número recibido por argumento en la línea de comando).

Ejercicio 5.6 (TP2 ejercicio 12)
Escriba un programa que le permita ingresar en arrays bidimensionales los apellidos y nombres de alumnos y las seis calificaciones obtenidas en cada parcial. Haga uso de una función que resuelva el ingreso, y devuelva cuántos se cargaron. Se garantiza que la cantidad de alumnos es menor que 100.
El array para las calificaciones tendrá una fila y una columna extra, en las que se calculará, haciendo uso de funciones al efecto:

- el promedio de calificaciones de cada alumno, que se acumula en la columna extra que le corresponde a cada alumno. Esta función debe ser invocada repetidamente con la dirección de cada fila del array.
- el promedio general de los alumnos para cada evaluación, y el promedio general.

Al terminar el programa debe generar una salida impresa (en archivo de texto), que debería verse así:



Note que:

- debe numerar las líneas de detalle,
- debe colocar un título indicativo, subrayarlo, y dejando un renglón en blanco (vacío), comenzar el listado de alumnos (hasta 21 por página),
- la numeración es consecutiva en las distintas páginas,
- los campos impresos deben quedar encolumnados,
- los promedios deben informarse en la última hoja,
- la primera línea se ha indicado para visualizar la alineación de los campos.

Ejercicio 5.7 (TP2 ejercicio 13)

Se dispone de dos archivos binarios: <empleados> y <estudiantes>.

Cada registro del primer archivo contiene los campos:

- <dni>, <apellido>, <nombre> y <sueldo>

en tanto que los del segundo:

- <dni>, <apellido>, <nombre> y <promedio>.

Ambos archivos están ordenados alfabéticamente por <apellido> / <nombre> / <dni>.

Ambos archivos deben ser leídos sólo una vez, y no deben ser almacenados en arrays. El sueldo es double y el promedio es float.

Escriba un programa que, leyendo ambos archivos, actualice el sueldo de aquellos empleados que tengan un registro de estudiante con un promedio superior a 7, en un 7,28%.

Ejercicio 5.8 (TP2 ejercicio 14)

Ingresar por teclado pares de cadenas de caracteres, finalizando el ingreso cuando ambas cadenas sean iguales (las que no deben procesarse). Para cada par, cargar en un array bidimensional, ambas cadenas, respetando cargar primero la más pequeña y luego la mayor, si las longitudes fueran iguales, el orden lo dará la comparación lexicográfica haciendo caso omiso de mayúsculas y minúsculas.

Escriba una función que determine la comparación solicitada invocando a versiones propias de las funciones de biblioteca estándar strlen y strcmpi o strcasecmp (dado que esta no es una función estándar en algunos compiladores tiene otro nombre).

Ejercicio 5.9 (TP2 ejercicio 15)

Escriba un programa que genere un archivo de texto de varias líneas, y en cada línea una o varias palabras (se considera palabra, cualquier carácter que no responde a lo indicado por la función de

biblioteca isspace, ni es coma, ni punto, ni ... (use su criterio). La separación entre palabras puede ser de uno o varios de estos caracteres. La primera palabra en una línea de texto puede estar precedida por más de uno de estos caracteres de separación, y lo mismo puede suceder después de la última palabra.
Escriba otro programa que agregue nuevas líneas de texto al archivo creado por el programa.
Escriba un programa que haciendo uso de su propia versión de strstr, busque en todo el archivo la subcadena recibida en la línea de comando (cualquiera sea esta), e informe en qué línea y en qué posición dentro de la línea la encuentra. Puede no encontrarla, encontrarla una única vez, o varias veces en la misma o distintas líneas del archivo.

Ejercicio 5.10 (TP2 ejercicio 16)
Leyendo el texto del archivo del ejercicio anterior, informar la cantidad de palabras que cumplen con cada una de las siguientes condiciones:

- están formadas por una sola letra,
- están formadas por una cantidad par de letras,
- comienzan con 'n',
- comienzan con un prefijo (o subcadena) determinado ingresado por el operador,
- tienen más de tres vocales,
- comienzan y terminan con vocales,
- contienen dígitos,
- sólo están formadas por dígitos,
- son palíndromos.

Ejercicio 5.11 (TP2 ejercicio 17)
Resuelva el <ej.: 13> con archivos de texto con campos de longitud fija.
Ídem, pero con archivos de texto con campos de longitud variable.
Note que deberá generar un nuevo archivo auxiliar de empleados con otro nombre, para al terminar eliminar (unlink) el archivo original y renombrar (rename) el auxiliar.

Ejercicio 5.12 (TP2 ejercicio 18)
Ingrese hasta un máximo de 1000 caracteres (o hasta que se ingrese el carácter '.'. Mostrar por pantalla el carácter inmediato posterior
(p. ej.: "a bgxj" -> "b!chyk"; "Zapato" -> "[bqbup"). Grabar en un archivo aquellas palabras con más de una determinada cantidad de letras que se deben pedir al operador al comienzo del proceso.

Ejercicio 5.13 (TP2 ejercicio 19)
Escriba una función:

- que le permita mostrar el contenido de un archivo binario de enteros cuyo nombre recibe por argumento. Cada registro se mostrará separado del siguiente por un espacio en blanco.
- que le permita mostrar el contenido de un archivo de texto cuyo nombre recibe por argumento. Cada registro se mostrará separado del siguiente por un espacio en blanco.
- que genere un archivo binario (<"archalea">) con 90 números al azar de tres cifras (función estándar rand de la biblioteca stdlib.h).
- que divida en tres nuevos archivos binarios (<"archal1">, <"archal2"> y <"archal3">) el

archivo anterior, tomado los primeros 30 para el primer archivo, los siguientes 30 para el segundo y los restantes para el tercero.

- que transforme en archivo de texto cada uno de los archivos del punto anterior (<”archal1.txt”>, <”archal2.txt”> y <”archal3.txt”>).
- que transforme los dígitos de cada número de los archivos del punto anterior en letras, generando los archivos de texto (<”archtex1.txt”>, <”archtex2.txt”> y <”archtex3.txt”>) de modo que a los dígitos del '0' al '9' les correspondan los caracteres de la ‘A’ a la ‘J’ para el primero, de la ‘K’ a la ‘T’ para el segundo y de la ‘U’ a la ‘D’ para el tercero. O sea:

		los dígitos se transforman										**Ejemplos**		
	Conversión	0	1	2	3	4	5	6	7	8	9	157	901	975
Para el archivo	1ro	A	B	C	D	E	F	G	H	I	J	BFH	JAB	JHF
	2do	K	L	M	N	O	P	Q	R	S	T	LPB	TKL	TRP
	3ro	U	V	W	X	Y	Z	A	B	C	D	VZB	DUV	DBZ

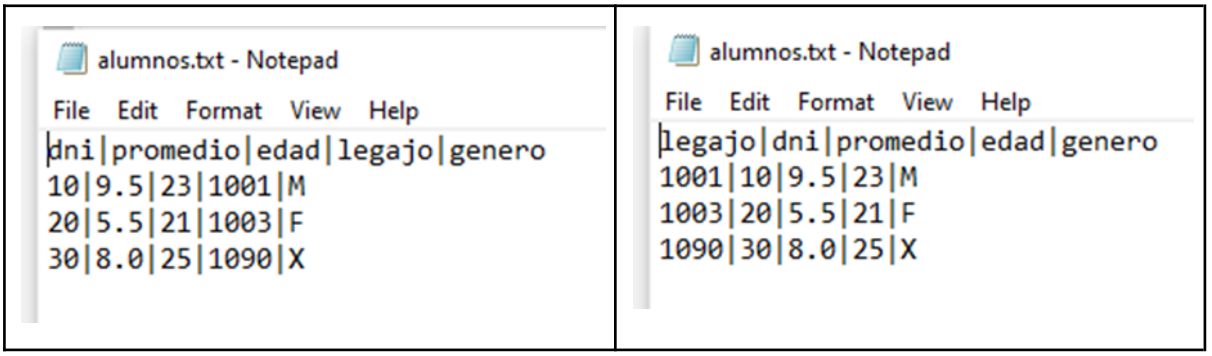
- que lea los archivos del punto anterior, y haciendo uso de la función <qsort> los ordene en forma creciente (únicamente aquí es necesario leer todo el archivo en un array). Se generan los archivos (<”archord1.txt”>, <”archord2.txt”> y <”archord3.txt”>).
- que genere un único archivo ordenado a partir de los archivos del punto anterior, manteniendo el ordenamiento ascendente. Se genera el archivo (<”archord.txt”>).

Escriba un programa que haciendo uso de cada una de las funciones (desde la tercera hasta la última), genere un archivo binario con 90 números al azar, lo separe en tres archivos, genere tres archivos de texto, etc.. Utilice las dos primeras funciones para visualizar el avance del proceso. Antes de terminar el proceso, utilice una función de menú para consultar si se desean eliminar los archivos intermedios.

Resuelva las funciones extra necesarias para una mejor estructuración del programa.

Ejercicio 5.14

Se tiene un archivo de texto con los campos “legajo, dni, promedio, edad, genero”, todos los campos son numéricos y no hay blancos entre ellos. La primera línea indica el orden de los campos en el archivo de texto y este puede cambiar entre distintos archivos, todas las combinaciones son posibles. Ej:



Desarrolle una función que genere un archivo binario de registros de la estructura tAlumno que se muestra al pie a partir de cualquier orden de campos en los archivos de texto. Utilice todas las técnicas aprendidas hasta la fecha para evitar poner una condición para cada orden de campos posible, recuerde que la cantidad de combinaciones es el factorial de la cantidad de campos, volviendo inconveniente esta estrategia

```
typedef struct{
    int dni;
    int legajo;
    float promedio;
    int edad;
    char genero;
}tAlumno;
```

Tenga presente que debe usar punteros a función para la resolución del ejercicio.

Unidad 6: Introducción a la Recursividad

Concepto de recursión. Estructura de funciones recursivas. Diferencias entre recursividad e Iteración.

Ejercicio 6.1 (TP2 ejercicio 20)
Escriba una función recursiva que:
- calcule el factorial de un número entero.
- muestre el contenido de un array de char.
- ídem anterior, mostrando en orden inverso.
- ídem anterior, devolviendo la suma de los caracteres que representan dígitos.
- muestre el contenido de un array de enteros en orden inverso, devolviendo la suma de todos los elementos.
- ídem anterior, devolviendo la suma de los pares.
- Ídem anterior, devolviendo la suma de los que están en posiciones pares.
- Escriba versiones recursivas de las funciones de biblioteca <strlen>, <strchr> y <strrchr>.

Ejercicio 6.2 (TP2 ejercicio 21)
El triángulo de Tartaglia es una forma sencilla de calcular los coeficientes de la potencia de un binomio. Valiéndose sólo de un array de enteros <unsigned short>, muestre por pantalla los primeros 19 juegos de coeficientes (potencias cero a 18).

Potencia	Coeficientes								
0	1								
1	1	1							
2	1	2	1						
3	1	3	3	1					
4	1	4	6	4	1				
5	1	5	10	10	5	1			
6	1	6	15	20	15	6	1		
7	1	7	21	35	35	21	7	1	
8	1	8	28	56	70	56	28	8	1
...	...								
18	…								

Compruebe que el siguiente juego de coeficientes (el que corresponde a la potencia 19) excederá el rango de almacenamiento de un entero corto. Diseñe nuevamente su cálculo para que se detenga cuando se dé esta condición, y no muestre los coeficientes en que se produce el problema. En los siguientes ejemplos falta omitir los coeficientes uno y las potencias cero.

$$(a+b)^0 = 1.a^0.b^0$$
$$(a+b)^1 = 1.a^1.b^0 + 1.a^0.b^1$$
$$(a+b)^3 = 1.a^3.b^0 + 3.a^2.b^1 + 3.a^1.b^2 + 1.a^0.b^3$$
$$(a+b)^6 = 1.a^6.b^0 + 6.a^5.b^1 + 15.a^4.b^2 + 20.a^3.b^3 + 15.a^2.b^4 + 6.a^1.b^5 + 1.a^0.b^6$$

Para ver mejor la tabla genere el archivo <tartaglia.txt>.