11.0 Trabajos Prácticos

TRABAJO Práctico I – Introducción a la programación

1- Supongamos que Ud. tiene una cuenta bancaria, una tarjeta que le permite el acceso a su cuenta a través de un cajero electrónico. Ud. realiza una operación de extracción de dinero de su caja de ahorros. Identifique (acorde con el modelo conceptual de Von Neumann) ENTRADA, PROCESO, SALIDA.

2- ¿Por qué razón Ud. cree que debemos utilizar un lenguaje de programación como C, BASIC, COBOL o JAVA para escribir un programa de computación? ¿Por qué no aprender directamente el lenguaje de máquina y evitar intermediarios?

3- Ud. desarrolla un programa para una empresa, dicha empresa le paga a Ud. únicamente por el servicio, la propiedad intelectual del programa le pertenece y Ud. no desea que el programa sea modificado sin su consentimiento. ¿Qué forma de ejecución de este programa le parece más apropiada: interpretado o compilado? Justifique.

4- Tache lo que no corresponda: "...como conclusión de la evolución histórica de los lenguajes de programación, podemos decir que los lenguajes más modernos tienen una estructura más acorde con el *razonamiento humano / funcionamiento interno del computador*, mientras que los primeros lenguajes tenían una estructura más acorde con el *funcionamiento interno del computador / razonamiento humano*...."

5- Responda verdadero (V) o falso (F) a las siguientes proposiciones:
La compilación me asegura una correcta ejecución del programa
Un programa que es interpretado se ejecuta más lento que si fuese compilado
Un programa compilado se ejecuta más lento que un programa interpretado
Un intérprete emite todos sus mensajes de error en tiempo de ejecución
Un programa compilado requiere del programa fuente para su ejecución
Un programa interpretado requiere del programa fuente para su ejecución

Marzo 2020

11.0 Trabajos Prácticos

TRABAJO Práctico II – Metodología de la Programación.

1. Unir con flechas (lenguajes de programación vs. paradigmas):

Cobol	Procedural
C	Funcional
C++	Lógico
Basic	Orientado a Objetos
Java	
Pascal	
Lisp	
Prolog	

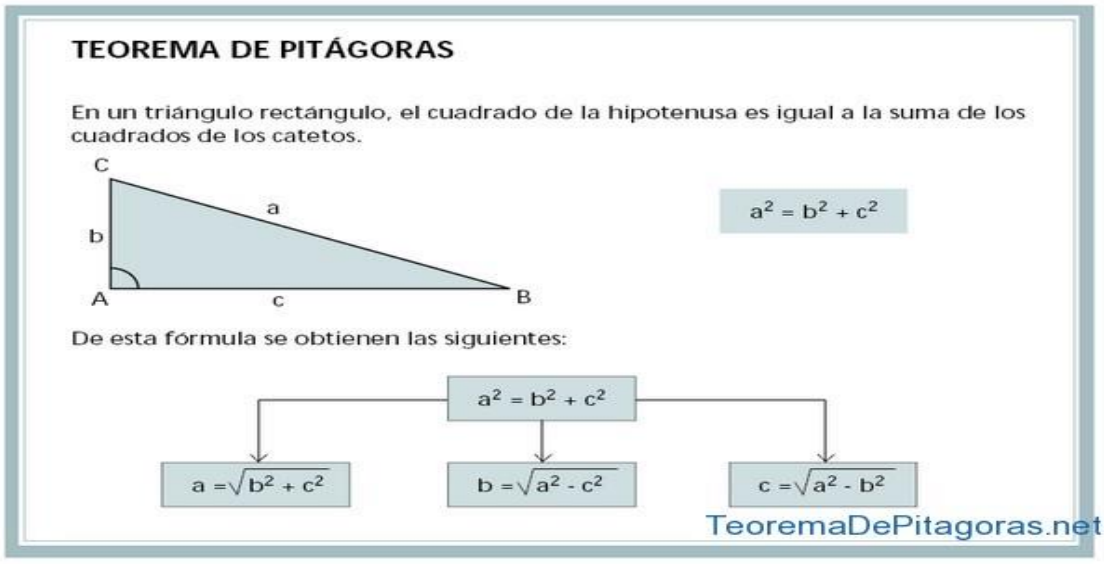
2. ¿Existe alguna relación entre el problema a resolver y el paradigma de programación a utilizar?
3. ¿Qué problemas puede ocasionar el “no documentar” los programas?
4. ¿Qué problemas puede ocasionar el resolver todos los problemas pensándolos algo indivisible, sin aplicar el concepto de modularidad?
5. Cuando hablamos de “código reutilizable” ¿a qué nos referimos con ello y cómo lo logramos?
6. ¿Para qué sirven los diagramas de flujo?
7. ¿Debemos hacer los diagramas de flujo de todos los programas que hagamos a lo largo de nuestra vida como programadores? Si – No ¿Por qué?

Marzo 2020

11.0 Trabajos Prácticos

TRABAJO Práctico III – Resolución de problemas de secuencia

1. Realice un algoritmo que permita ingresar un numero por teclado y mostrar el mismo por pantalla.
2. Realice un algoritmo que permita ingresar un numero por teclado, multiplicar el numero ingresado por 2 y mostrar el resultado por pantalla.
3. Realice un algoritmo que permita ingresar dos números por teclado y muestre por pantalla el resultado de la suma de ambos números.
4. Realice un algoritmo que permita ingresar dos números por teclado y muestre por pantalla el resultado de la suma y de la resta de ambos números..
5. Realice un algoritmo que permita ingresar dos números por teclado y muestre por pantalla el resultado de la suma, la resta, la multiplicación y la división.
6. En EEUU, las temperaturas se expresan en grados Fahrenheit, la fórmula para pasar a grados Celsius (Centígrados) es: Grados Celsius = (5 / 9) * (Grados Fahrenheit – 32). Realice un algoritmo para que un viajero pueda ingresar grados Fahrenheit y muestre en pantalla el equivalente en grados celsius.
7. Realice un algoritmo similar anterior, pero esta vez, en vez de ingresar grados Fahrenheit se ingresan por grados centígrados y se muestra en pantalla el equivalente en grados Fahrenheit.
8. Realice un algoritmo para calcular la superficie de un triangulo, en donde la formula es (base * altura) / 2.
9. El teorema de Pitágoras dice que en un triangulo rectángulo, el cuadrado de la hipotenusa es igual a la suma de los cuadrados de los catetos. Dado el triangulo ABC podemos decir:



-realice un algoritmo que permita ingresar el valor del cateto b y del cateto c y muestre en pantalla el valor de la hipotenusa (a)
-realice un algoritmo que permita ingresar el valor de la hipotenusa (a) y del cateto c y muestre en pantalla el valor del cateto b
-realice un algoritmo que permita ingresar el valor de la hipotenusa (a) y del cateto b y muestre en pantalla el valor del cateto c
Verifique los resultados de su algoritmo en http://teoremadepitagoras.info/teorema-de-pitagoras-calcular-hipotenusa-o-catetos

11.0 Trabajos Prácticos

TRABAJO Práctico III – Resolución de problemas secuencia

10. Realice un algoritmo para ingresar un numero y mostrar su raiz cuadrada.
11. Realice un algoritmo para ingresar un numero decimal y muestre por pantalla la parte del número y luego la parte decimal (al menos hasta 2 digitos decimales). Ejemplo: se ingresa 123.35 y como salida debe mostrar 123 y luego 35.
12.

Traduzca cada uno de los diagramas de flujo en programas ANSI C, compile y ejecute el código resultante, verifique el correcto funcionamiento de cada programa.

Marzo 2020

11.0 Trabajos Prácticos

TRABAJO Práctico IV – Resolución de problemas de selección

1. Verifique el algoritmo que mostraba la suma, la resta, la multiplicación y la división de dos números ingresados por teclado. Este algoritmo debería fallar si el segundo numero es igual a cero; cuando se intente hacer la división utilizando el cero, debería dar error, no puede resolverse una división por cero. Modifique este algoritmo para que, si el usuario ingresa un cero en el segundo número, emita un mensaje de error y termine; caso contrario, que continúe con los cálculos.
2. Realice un algoritmo que permite ingresar dos números cualquiera y emite un mensaje como salida indicando cual de los dos numero es mas grande y cual es mas pequeño.
3. Realice un algoritmo que permita ingresar un numero y emita un mensaje indicando si el numero es entero o decimal, según corresponda.
4. Realice un algoritmo que permita ingresar un número e indique si el mismo es par o impar.
5. Realice un algoritmo que permita ingresar un número y determine si el cubo del numero es positivo o negativo.
6. Realice un algoritmo que permita ingresar dos números y mostrar un mensaje indicando si el primer número es divisible por el segundo o no.
7. Realice un algoritmo que permita ingresar 3 números: los dos primeros números indican un rango y el algoritmo emitirá un mensaje indicando si el tercer número se encuentra dentro del rango o no. Ejemplo: se ingresa: 100,200,300 el mensaje indicará que 300 esta fuera del rango (100-200). Otro ejemplo: se ingresa: 100,200,150 el mensaje indicará que 150 se encuentra dentro del rango (100-200, en este caso).
8. El algoritmo de conversión de grados Farenheit a Celsius, permite ingresar valores Fahrenheit no lógicos, tales como 1000 grados F. Ahora este algoritmo sólo aceptará grados Fahrenheit entre -100 y 140. Para cualquier valor ingresado fuera de este rango, emitir un mensaje de error indicando que no es posible ingresar este valor, mostrando el rango de valores permitidos y terminar el programa.
9. Realice un algoritmo que permita ingresar por teclado los 3 lados de un triángulo (a, b, c) e imprima por pantalla un mensaje indicando qué tipo de triángulo es (isósceles, escaleno o equilátero). Recordemos que un triangulo es isósceles tiene 2 lados iguales y uno distinto; el escaleno tiene todos los lados distintos y el equilátero tiene todos los lados iguales.
10. Realice un algoritmo que permita ingresar por teclado un numero entero (entre 0 y 255 ambos inclusive) y muestre en pantalla su equivalente en numeración binaria.

Traduzca cada uno de los diagramas de flujo en programas ANSI C, compile y ejecute el código resultante, verifique el correcto funcionamiento de cada programa.

Marzo 2020

11.0 Trabajos Prácticos

TRABAJO Práctico V – Resolución de problemas de iteración

1. Realice un algoritmo que sirva para sumar los números que se ingresen por teclado. A medida que se ingresan los datos, se debe ir mostrando el total acumulado. El programa termina de ejecutarse cuando se ingresa el valor 0 (cero).
2. Realice un algoritmo que sirva para ingresar números (el programa deja de pedir números por teclado cuando se ingresa el valor 0 (cero)) y al terminar el ingreso de números muestre: la sumatoria y el promedio.
3. Copie el algoritmo anterior. Ahora es el usuario quien indica previamente cuántos números va a ingresar. Luego de ello se ingresa la cantidad de números indicada y se emite la misma salida que en el algoritmo anterior: la sumatoria y el promedio.
4. Copie el algoritmo anterior y modifíquelo para que, además de la sumatoria y el promedio; también muestre el valor más pequeño ingresado y el valor más grande ingresado.
5. Copie el algoritmo que hacía cálculos de suma, resta, multiplicación y división (tp iv ejercicio 1) y modifíquelo para que una vez que éste ha mostrado el resultado de los cálculos, el programa vuelva a solicitar datos y continúe haciendo cuentas (en vez de terminar la ejecución). Cuando el usuario ingrese un cero (0) en el primer operador y un cero (0) en el segundo operador, esto será señal de que el usuario desea terminar. Recuerde que debe tener una "salida del algoritmo", sino el usuario quedará "eternamente" dentro de un ciclo infinito de ejecución del programa.
6. De la misma forma que en el caso anterior, copie y modifique el algoritmo de conversión de grados Fahrenheit (tp iv ejercicio 8) para que vuelva a pedir el ingreso de grados Fahrenheit y continuar haciendo conversiones. Cuando el usuario ingrese 999 grados Fahrenheit, será señal de que desea dejar de usar el algoritmo.

Ejercicios con Arreglos

7. La Empresa Y posee 10 sucursales. Realice un algoritmo que permita ingresar el total de ventas de cada sucursal, luego imprima las ventas previamente ingresadas.
8. Copie y modifique el algoritmo anterior, para que además imprima el total general de ventas.
9. Copie y modifique el algoritmo anterior. Ahora la Empresa Y tiene 3 sucursales y 5 vendedores en cada sucursal. Ingrese por teclado las ventas totales diarias de cada vendedor en cada sucursal (utilice para ello un arreglo bidimensional de 3 filas por 5 columnas).
10. Copie y modifique el algoritmo anterior. El algoritmo deberá mostrar los totales por cada sucursal (para todos sus vendedores, para ello utilice otro arreglo de 3 elementos en donde sumarizar las ventas de todos los vendedores de cada sucursal). Por último, muestre el total general de ventas (utilice una variable como acumulador).
11. Copie y modifique el algoritmo anterior. Ahora la Empresa Y vende en todo el país. El país ha sido dividido en 3 zonas, en cada zona hay 4 sucursales y en cada sucursal hay 2 vendedores. Ingresar los totales de ventas de cada zona, de cada sucursal y cada vendedor. Utilice un arreglo de 3 dimensiones para almacenar las ventas.
12. Copie y modifique el algoritmo anterior para emitir totales de ventas por sucursal, por zona y total general.
13. Copie y modifique el algoritmo anterior, para hacerlo más flexible, ahora es el usuario del programa quien indica cuántas zonas hay, cuántas sucursales por zona y cuántos vendedores por sucursal. Ingresar estos datos, validarlos. Ingresar las ventas que correspondan y emitir los totales por sucursal, por zona y general.

Traduzca cada uno de los diagramas de flujo en programas ANSI C, compile y ejecute el código resultante, verifique el correcto funcionamiento de cada programa.

Marzo 2020

11.0 Trabajos Prácticos

TRABAJO Práctico VI – Resolución de problemas utilizando arreglos, caracteres, memoria dinámica

1. Implemente una función que devuelva un texto ingresado por teclado (`char *`). Dicho texto se ingresa por teclado utilizando la función getchar() . Guarde cada uno de los caracteres ingresados en un buffer de caracteres de una longitud fija y máxima que Ud. considere como límite para esta función. La función deberá retornar los caracteres ingresados cuando el usuario presione la tecla Enter (Intro, `'\n'`) o bien cuando se halla alcanzado el límite máximo de caracteres fijados. No olvide del caracter `'\0'` final que require toda cadena de caracteres. Ante cualquier error, se deberá devolver `NULL`.
2. Modifique la función anterior, de forma tal que ésta ya no tenga ningún límite en cuanto a la cantidad de caracteres ingresados por teclado. Para ello se requiere solicitar memoria dinámica (`malloc(), realloc(), calloc()`) a medida que sea necesario, para dar lugar a más cantidad de caracteres a ingresar (`realloc()`), hasta que termine el ingreso por teclado. No pida memoria "de a 1 caracter", ello sería muy ineficiente, hágalo "de a bloques" de caracteres múltiplo de potencias de 2, ejemplo: 256, 512, 1024, etc. Ante cualquier error, se deberá devolver `NULL`.
3. Implemente otra función para el ingreso de caracteres por teclado, usando la misma técnica que en el punto 1, pero esta vez se le indica a la función un máximo de caracteres a ingresar, si el usuario ingresa más caracteres, éstos serán ignorados. Ante cualquier error, se deberá devolver `NULL`.
4. Implemente una función que permita ingresar un conjunto de líneas de texto ingresadas por teclado; la función recibe como argumento la cantidad máxima de líneas de texto a ingresar y un arreglo de punteros a caracter (`char*[]`) que contendrá todas las líneas ingresadas por teclado y asignadas dinámicamente (por lo tanto, las otras funciones que la utilicen serán responsables de liberar la memoria de este arreglo una vez que el mismo ya no se requiera (`free()`)) a su vez, reutilizará, usará la función del punto 2. La función devolverá el número de lineas efectivamente ingresadas (un número menor o igual al máximo de líneas a ingresar).
5. Implemente una función que permita saber: a) ¿cuál fué la cadena de caracteres más corta ingresada? b) ¿cuál fue la cadena más larga? c) ¿cuál es la posición (en el arreglo de cadenas) de la cadena más corta? d) ¿cuál es la posición de la cadena más larga? . La función recibirá como argumentos de entrada un arreglo de punteros a cadenas (char*[]) , es decir, el arreglo usado como argumento de la función del punto 4; recibirá también la cantidad máxima de cadenas ingresadas; como no es posible devolver más de un valor, entonces la función no devolverá nada (`void`). Entonces: ¿Cómo podemos hacer para devolver las respuestas a), b), c) ,d) ? . Discútalo con el profesor.
6. Reutilice las funciones anteriores en un programa que permita el ingreso de líneas de texto por teclado, muestre en pantalla la línea más pequeña ingresada y la línea más larga ingresada. Pruebe su correcto funcionamiento. Libere la memoria utilizada de forma correcta, haga n repeticiones (`for()`) de esto para asegurarse de no tener problemas de memoria.
7. Realice un simple programa para desplegar un menu de opciones en la consola. Luego modifíquelo para convertirlo en un “menú genérico”, reusable en sus distintos proyectos. Se puede “encapsular” todo el menú de la siguiente forma: crear la función int menu(char *titulo,char *opciones[]) esta función devuelve la opción elegida por el usuario, “encapsula” toda la lógica del menú (iteración, mensajes de error, títulos, ingresos de datos, etc.), recibe como argumento el título del menú y un arreglo de tipo char * que contendrá las opciones del menú (sólo las opciones, sin los números, sin la opción “Salir”, este arreglo termina con NULL). La forma de uso sería:

```
char *titulo = "Menu de Prueba";
char *arrayOpciones[] = { "Opción 1", "Opción 2", "Opción 3", NULL };
int opcion;
opcion = menu(titulo,arrayOpciones);
```

11.0 Trabajos Prácticos

TRABAJO Práctico VI – Resolución de problemas utilizando arreglos, caracteres, memoria dinámica (continuación)

8. Realice una función que reciba como entrada un buffer de tipo `char *` y devuelva como retorno la cantidad de líneas (`\n`) que hay en el mismo. Suponiendo que dicho buffer tiene la forma de un archivo de texto: `<<texto>>\n<<texto>>\n\n...\n\0`
9. Realice una función que reciba como entrada un buffer de tipo `char *` (que representa el contenido de un archivo de texto, como el indicado en el punto anterior) y un arreglo de tipo `char *[]` (que representa un arreglo de punteros a cada una de las lineas de texto del buffer `char *` pasado como primer argumento). La función deberá parsear el primer argumento e ir creando dinámicamente líneas de texto a ser apuntadas por el segundo argumento. La función no devuelve nada (`void`), sólo realiza este proceso y termina. Tener en cuenta que luego de la ejecución de esta función, existirá en memoria el buffer pasado como primer argumento más las nuevas líneas creadas.

Marzo 2020

11.0 Trabajos Prácticos

TRABAJO Práctico VII – Búsquedas, Ordenamiento, Modularidad, Reutilización, Creación de Librería

1. Copie de la pagina web de la asignatura las carpetas `/Ejercicios/search` , `/Ejercicios/sort` ; a su directorio de trabajo `/src` , ingrese a cada nueva subcarpeta creada, abra cada programa `.c`, observe su contenido, investigue todas las llamadas a funciones que no conozca (por ejemplo, en `/Ejercicios/sort/quick/quick.c`: `srand()`, `getpid()`, `rand()`, realice su compilación y luego ejecútelos para comprobar su correcto funcionamiento. En el caso de que no conozca alguna sentencia (por ejemplo, en `/Ejercicios/sort/quick/quick3struct.c`: `struct ... {...}`) solicite la explicación del profesor.
2. Cree un nuevo programa (`quickline.c`) a partir de `/Ejercicios/sort/quick/quick3struct.c` para que pueda ordenar un arreglo de punteros a cadenas de caracteres (char *[]), como los utilizados en los puntos 4-7 del tp VI.
3. A partir del código provisto en `/Ejercicios/sort/quick` , el programa `quickline.c`, y cualquier otra función adicional que Ud. pretenda realizar, teniendo en cuenta los conceptos de modularidad y reusabilidad, teniendo en cuenta lo indicado en `/Apuntes/ ProgramacionI-2008-gcc.doc` sección "`Compilación y Catalogación de Librerías`"; genere una librería que permita el ordenamiento de arreglos de `int []`, `double []`, `char *[]`, `char *`, etc. utilizando el algoritmo quick-sort. Genere un archivo de cabecera para dicha librería en `/include` , tenga en cuenta la interfase de usuario que tendrá la librería, genere un documento de ayuda que permita utilizar dicha librería en `/doc` , genere el archivo `.a` en `/lib`. Realice algun programa de ejemplo del uso de esta librería. Solicite ayuda y orientación al profesor, así como también consejos acerca de cómo generar la documentación de la librería.
4. Modifique el programa `menu` del tp VI, más precisamente, la función `int menu(char *titulo,char *opciones[])` , para que ahora haga uso de la librería creada en el punto anterior y antes de mostrar las opciones de menú, haga un ordenamiento de las mismas.
5. Cree un nuevo programa (`binaryall.c`) a partir de `/Ejercicios/search/binary/binary.c` para hacer nuevas versiones de la función `binary_search()` de búsqueda binaria que permita buscar en arreglos de enteros `int []`, `double []`, arreglos de cadenas de caracteres `char *[]` , etc.. De forma tal, de darle la mayor usabilidad posible a estas funciones. Utilice la función `main()` para probar las nuevas funciones creadas a partir de `binary_search()`.
6. Una vez probadas las funciones de búsqueda del punto anterior, incorpore el nuevo código a la librería creada en el punto 3, de forma tal de transformar a esta librería en una valiosa herramienta de búsqueda y ordenamiento a ser utilizada en cualquier nuevo proyecto. Actualice los archivos de cabera, el `.a`, y la documentación de la librería. Amplíe el programa de ejemplo para el uso de esta librería. Solicite ayuda y orientación al profesor.

Marzo 2020

11.0 Trabajos Prácticos

TRABAJO Práctico VIII – Trabajo Práctico Final Integrador

Este trabajo es individual (a diferencia de los anteriores) y debe ser consensuado con el docente. Se trata de implementar un proyecto (lo cual implica que Ud. debe contar con un ambiente de trabajo como el indicado en el documento "/Apuntes/ProgramacionI-2008-gcc.doc" sección "Nuestro Ambiente de Trabajo"), dicho proyecto debe reunir las siguientes características:

- Se trata de una aplicación de consola o bien en 3 capas CGI (no requiere implementar GUI en C)
- Puede ser una aplicación interactiva, en tal caso, debe contar con un menu de opciones y mensajes entendibles por el usuario para su correcta operación
- Puede ser una aplicación no interactiva, en tal caso, puede utilizar la línea de comandos como forma de comunicación con el exterior, STDIN, STDOUT, STDERR, en tal caso, cuando no se indiquen argumentos (o bien cuando se indique el argumento -help), el programa deberá indicar su forma de funcionamiento
- La aplicación debe combinar distintas estructuras repetitivas
- La aplicación debe tener una utilidad práctica comprobable, mejor aún si se trata de un proyecto real
- Se debe tener presente la funcionalidad que se pretende lograr y las futuras versiones que podrían realizarse de la misma
- Es deseable que maneje algún tipo de información persistente
- Es deseable -en caso de ser necesario- que se aplique alguna estructura de datos de alto nivel
- Deberá contar con alguna forma de búsqueda u ordenamiento si el problema lo permite
- La aplicación debe ser modular, en lo posible, puede adjuntarse una o más librerías propias desarrolladas en paralelo con la aplicación y que ésta utilice
- Todos los programas deben estar correctamente documentados e identados desde los programas fuentes utilizado Doxygen (http://www.doxygen.org) u otra herramienta similar, generando toda la documentación en formato html.
- Se debe adjuntar un documento indicando: la especificación del programa, instalación, diseño, entradas, procesos, salidas, manual del usuario.
- Entregar en soporte electrónico el ambiente de trabajo, la documentación y la aplicación compilada lista para ser instalada.

El proyecto deberá estar terminado (por lo menos en su primera versión) 15 días antes de la finalización de la cursada para su revisión, pasado este plazo, deberá acordar con el Profesor los plazos de entrega del mismo. Se acuerda la siguiente versión del proyecto que deberá estar lista para la fecha de final en que se presente el alumno, ya que será parte de su evaluación final.
Si el alumno no sabe que tema desarrollar o que aplicación realizar, le aconsejamos que lea el TP XI Temas de Investigación y acuerde con el docente el trabajo a realizar.

Marzo 2020

11.0 Trabajos Prácticos

TRABAJO Práctico IX (Opcional) – Persistencia

Realice una función que permita leer cualquier tipo de archivo (tenga en cuenta que su contenido podría ser binario) y cargue su contenido en memoria. Puede utilizar un buffer de tipo void `*`, asignado dinámicamente, combinado con una variable de tipo `long` que indique la cantidad de caracteres leidos. Utilice una lectura tipo "buffered", usando las funciones: `fopen()`, `fread()`, `ftell()`, `fseek()`, `fclose()`.

Realice la función inversa al punto anterior, es decir, dado un buffer void * cargado en memoria y una variable de tipo `long` que indica la cantidad de caracteres del mismo, junto con un nombre de archivo, permita grabar dicho contenido de memoria en el archivo indicado. Si el archivo indicado existía previamente, deberá ser sobreescrito. El archivo podría tener contenido binario. Utilice `fopen()`, `fclose()`, `fwrite()`.

Supongamos un negocio cuyo sistema informático va registrando los artículos que se venden (a medida que se venden en el mostrador), junto con sus cantidades y precio (al momento de la venta). Dicha información se guarda en un archivo de ventas. Dicho archivo es secuencial, su contenido es de texto (no binario) y cada registro es delimitado por `\n` (una línea de texto por cada artículo que se vende), por lo tanto, es un archivo "editable/visualizable" por cualquier editor de texto (ej.; `notepad.exe`, `vi`, `edit`, `emacs`, etc.). Los campos no estan delimitados, sino que se graban en determinadas posiciones fijas dentro de cada registro: posición 0 código artículo (máximo 10 caracteres-números), posición 12 cantidad (máximo 5 dígitos + punto + 2 decimales), posición 22 precio (máximo 6 dígitos + punto + 2 decimales). Realice un programa que permita el ingreso de estas ventas por teclado y guarde cada venta en el archivo `ventas.dat` , el usuario deja de ingresar cuando presiona `Intro` en el código de artículo. Utilice `notepad.exe` para verificar que `ventas.dat` ha sido creado correctamente. Cada vez que se ejecute este programa, si `ventas.dat` existe, el mismo deberá continuar agregando ventas (sin perder las ventas previamente guardadas). Puede utilizar la función `fprintf()`, teniendo el cuidado de no desplazar las posiciones de los campos, caso contrario, utilizar un buffer de tipo `char *` y "llenar" allí cada registro a grabar utilizando la función `fwrite()`.

Realice una función que utilice la función del punto 1, lea una archivo de texto cualquiera (puede utilizar al archivo `ventas.dat` generado en el punto anterior) y permita su visualización en pantalla de a 20 líneas, emita un mensaje en pantalla preguntándole al usuario si desea continuar o no con la visualización (`Intro` implica continuar con la visualización, cualquier otra tecla implica salir). Debe ser un programa al estilo del programa `more`. Consulte a su profesor.

Marzo 2020

11.0 Trabajos Prácticos

TRABAJO Práctico IX (Opcional) – Persistencia (continuación)

Realice un programa que tenga como entrada al archivo `ventas.dat` y emita como salida el archivo `ventas.cla` . Ambos archivos contienen la misma información y tamaño, pero `ventas.cla` está ordenado por código de artículo (recuerde que dicho código se podía repetir en la medida en que haya varias ventas del mismo artículo durante el día). Utilice la función del punto 1 para cargar a `ventas.dat` en memoria, luego utilice las funciones creadas en los puntos 8 y 9 del tp VI para realizar un "parsing" de lo cargado en memoria; luego utilice la librería creada en el tp VII para ordenar el buffer de tipo `char *[]` (que representan a cada una de las líneas de `ventas.dat`) de memoria y luego grabe su contenido al disco (`ventas.cla`) con las líneas ordenadas. Libere correctamente la memoria asignada dinámicamente. Verifique este algoritmo y consulte con el profesor.

Realice un programa que genere un archivo de texto secuencial (`ventas.dat.listado1`), a modo de un listado, que contenga el título "Total Ventas por Artículo", a dos columnas: Artículo, Cantidad. La primer columna indica los códigos de artículos (deberán ser todos distintos), la segunda es la sumatoria de las cantidades vendidas. Este programa recibe como entrada a `ventas.cla` del punto anterior y realiza un "corte de control" por código de artículo sobre dicho archivo. Solicite la explicación del profesor en cuanto a cómo implementar un algoritmo de "corte de control" y qué relación tiene éste con las técnicas de ordenamiento.

Idem anterior, realice un nuevo programa a partir del anterior para generar un archivo de texto secuencial (`ventas.dat.listado2`) igual que el anterior, pero ahora agregando dos columnas más: Precio Promedio (precio promedio de venta de cada unidad del artículo) y Total $ (sumatoria de cantidad por precio de venta, de cada una de las ventas de dicho artículo). Al final se deberá agregar al listado un Total General de $ vendidos.

Parametrice -a través de la línea de comandos- los programas de los puntos 4, 5, 6, 7 de forma tal de indicar la entrada y salida (cuando corresponda) de cada uno de ellos desde la línea de comando, de forma tal, que los nombres de los archivos de entrada o salida se indiquen desde el exterior del programa. Implemente un archivo batch de windows o bien un archivo .sh del bash shell de linux/cygwin, para realizar una ejecución batch del programa del punto 5, luego el 6, luego el 7, luego el 4 para visualizar el primer listado y nuevamente el 4 para visualizar el otro resultado.

Realice un programa que permita hacer un "corte de control" sobre el archivo `ventas.cla`, realizando la sumatoria de cantidades vendidas de cada artículo y descuente dicha cantidad del archivo maestro de productos. Consulte con el profesor acerca de cómo debería estar implementado el archivo maestro de productos.

Marzo 2020

11.0 Trabajos Prácticos

TRABAJO Práctico X (Opcional) – Estructuras de Alto Nivel

Copie de la pagina web de la asignatura las carpetas `/Ejercicios/data structures`; a su directorio de trabajo `/src` , ingrese a cada nueva subcarpeta creada (excepto `/code`) abra cada programa `.c`, observe su contenido, investigue todas las llamadas a funciones que no conozca, realice su compilación y luego ejecútelos para comprobar su correcto funcionamiento. En el caso de que no conozca alguna sentencia, solicite la explicación del profesor.

En el archivo maestro de productos que se mencionaba en el punto 9 del tp IX, se guardan todos los datos relativos a cada producto de la empresa (código de producto, descripción, precio de venta, stock). Realizar un programa que permita hacer altas, bajas, modificaciones y consultas sobre dicho archivo. Analice con el profesor la mejor forma de implementar un archivo relativo para este propósito.

Modifique el programa anterior, para agregar la siguiente funcionalidad: ahora nos interesa acceder a la información de los productos a través de la primer letra de la descripción: por ejemplo,si el usuario ingresa una 'P' el programa deberá listar la dirección relativa, código, descripción, precio y stock de todos los artículos cuya descripción comience con la letra 'P'. Facilitando la ubicación de los productos cuando se desconoce el código y/o dirección relativa. Analice con el profesor una forma posible de implementación de este algoritmo utilizando una lista enlazada.

Modifique el programa anterior para agregarle una opción de menú que permita listar (guardando la salida en un archivo de texto) alfabéticamente los productos del stock.

En la Empresa Y todos los empleados tienen -acorde con sus funciones- un programa de capacitación permanente que consta de cursar y aprobar una serie de cursos. Cada curso tiene un código unívoco (C1, C2, C3, ...), una nota o calificación obtenida por el empleado (número entre 0 y 10) y la fecha de dicha nota o calificación. Los datos que importan del empleado son: número de legajo (número entero de 5 dígitos), apellido, nombres. Si un empleado desaprueba un curso, tiene que volver a realizarlo en el futuro, por lo tanto, un mismo curso puede estar asociado más de una vez con un empleado determinado. Realizar un programa que permita hacer altas, bajas, modificaciones de empleados y de cursos asociados a empleados. Diseñe una interfase lo más cómoda posible para el usuario. Consulte con el profesor cómo implementar este algoritmo -en principio- en memoria.

Modifique al programa anterior para dotarlo de persistencia, es decir, que los datos no se pierdan y tengan que volver a ingresarse por cada ejecución del programa. Consulte con el profesor cómo implementar este algoritmo.

Marzo 2020

11.0 Trabajos Prácticos

TRABAJO Práctico XI (Opcional) – Temas de Investigación

Ver http://www.grch.com.ar/docs/p1/Investigacion/

1. Investigar acerca del uso y utilización de la librería pthread disponible en cygwin y gcc, hay documentación de la misma en la pagina web de la asignatura y ejemplos de programas que la utilizan. Esta librería permite una manera portable de implementar multitarea (multithreading) a nuestros programas.
2. Investigar acerca de la forma de conectar un programa C a una base de datos desde cygwin o gcc, utilizando una librería que permita tal empresa.
3. Investigar acerca de networking, programas que intercambian datos por la red, la librería para el manejo de sockets tcp, hay documentación de la misma en la pagina web de la asignatura y ejemplos de programas que la utilizan. Desarrollo de servidores y clientes.
4. Se pueden combinar los puntos 1, 2, 3 en un único proyecto.
5. Investigar acerca de interfases gráficas utilizando C, se propone a) un camino "propietario" orientado a interfaces gráficas sólo para Windows a través del uso de la API (application program interface) de windows ; b) otro camino "propietario" a través del uso de las librerías x11R6, x11R5, etc. para el desarrollo de aplicaciones xwindows sólo para Unix/Linux ; c) un camino portable a través del uso de librerías tales como GTK+ que permitan el desarrollo de GUI (graphical user interfaces) portables a Windows, Unix/Linux.
6. Investigar acerca del desarrollo de aplicaciones orientadas a caracter utilizando la libreria ncurses, la cual permite hacer un manejo de ventanas y posicionamiento por filas y columnas dentro de la consola.
7. Investigar acerca de la interface CGI (Common Gateway Interface) que permite el tratamiento de formularios HTML (hyper text markup language) en un servidor web, permitiendo la ejecución de programas C para el tratamiento de estos formularios y devolviendo código HTML, permitiendo la realización de páginas HTML dinámicas. Esto implica un desarrollo de aplicaciones moderno, orientado a internet, sistemas de 3 capas: capa cliente o de presentación (formulario html), capa de aplicaciones (web server que ejecuta proceso cgi) y capa de datos (archivos o base de datos).
8. Investigar acerca de XML como nuevo lenguaje de intercambio de datos y como utilizar estos documentos en el lenguaje C a partir de la utilización de la librería libxml2, esta librería esta disponible en cygwin y gcc, en la pagina web de la asignatura hay ejemplos de código utilizando esta librería.
9. Idem anterior con los archivos comprimidos .zip utilizando la libreria zlib, librería general para la compresión de archivos.
10. Unirse al proyecto SGANS y desarrollar la parte CGI del mismo utilizando Servidor Mongoose y librería libfb, libcgi. Ver http://www.grch.com.ar/docs/pp/2012/sgans/tpfinal.pdf Ver http://sourceforge.net/projects/libfb
11. Leer los capítulos 4 y 5 del libro Sistema de base de datos de Elsmari-Navathe que se encuentra en la carpeta: http://www.grch.com.ar/docs/bd/libros allí se explican detalladamente varias organizaciones de archivos, las cuales se podrían implementar en lenguaje C.

Marzo 2020