

Universidad de Panamá

Centro Regional Universitario de Veraguas

Facultad de Informática, Electrónica y Comunicación

Inf212 – Programación II y Programación III – Inf200

Tarea de Desempeño #2 - Uso de matrices

Integrantes:

Aguilar, Faustino

Concepción, Andrés

Facilitador: Profesor Diego Santimateo

Abril, 2015

DESCRIPCIÓN DE LA TAREA

Se trata de diseñar y codificar un programa utilizando el lenguaje de programación C, el cual, a partir de dos matrices debe calcular el salario bruto de determinados empleados (se deben indicar los nombres de los empleados cuyos salarios serán calculados) y el total en dinero de la planilla de los empleados solicitados. El programa maneja dos matrices una de nombres de empleados y otra que contiene en una columna las horas trabajadas y en la otra columna el salario por hora. Puede asumir que el número de la fila que contiene el nombre es el mismo que contiene las horas trabajadas y el salario por hora.

Algoritmo necesario para obtener la salida:

- Leer **n** nombres, colocar cada nombre en la matriz de nombres y en la misma fila de la otra matriz, colocar las horas trabajadas y el salario por hora. Observe que se usa la misma fila en ambas matrices para la misma persona. No lo olvide.
- Cuando termina de cargar sus matrices y en un nuevo ciclo... lea **m** nombres, busque cada nombre en la matriz de nombre, si lo encuentra, suspenda la búsqueda y acceda a las horas trabajadas y salario por hora de la otra matriz usando el mismo índice de fila donde encontró el nombre. Debe indicar si el empleado no existe.
- Multiplique horas trabajadas * salario por hora para lograr el salario bruto de cada empleado.
- Acumule los salarios calculados.

INVESTIGACIÓN PREVIA

Concepto de arreglo: es una estructura de datos estática que almacena elementos de manera ordenada y de tipo unificado.

Clasificación: los arreglos pueden ser unidimensionales como los vectores o multidimensionales como las matrices.

Formas de acceder: se pueden manejar los elementos de un arreglo con su índice que no es más que la ubicación del elemento dentro del arreglo. El índice del primer elemento del arreglo es cero (0).

Tipos de elementos: los arreglos solo pueden ser de un tipo a la vez, es decir que un arreglo solo puede ser *char* (cadena) o solo puede ser *int* (entero). Los arreglos de tipo cadena tienen como último elemento el carácter nulo (`\0`).

Presentación de ejemplos de arreglos y sus elementos: para identificar los arreglos tenemos los siguientes ejemplos:

<code>int vector[10];</code>	Arreglo unidimensional de tipo entero.
<code>float matriz[5][5];</code>	Arreglo bidimensional de tipo flotante.
<code>char arreglo[2][4][3];</code>	Arreglo multidimensional de tipo cadena.
<code>int arreglo[2][2][2];</code>	Arreglo multidimensional de tipo entero.

Además tenemos conocimientos de los ejemplos que practicamos en clase disponibles en nuestro repositorio de código: http://github.com/nodetino/arreglos_c

Funciones de la librería de C: para la realización del proyecto utilizamos las siguientes librerías:

stdio.h

Librería de entrada y salida que nos permite imprimir en pantalla y leer datos del usuario. Las principales funciones que utilizamos son *printf* y *scanf*.

stdlib.h

Librería de utilidades del sistema, nos permite ejecutar comandos del sistema operativo. En este caso utilizamos la función *system*.

string.h

Librería para manejar cadenas de caracteres, en este caso utilizamos las funciones *strcmp* para comparar cadenas de caracteres, retorna cero (0) si son iguales.

Fuentes de información consultadas: para la realización de este proyecto consultamos las siguientes fuentes:

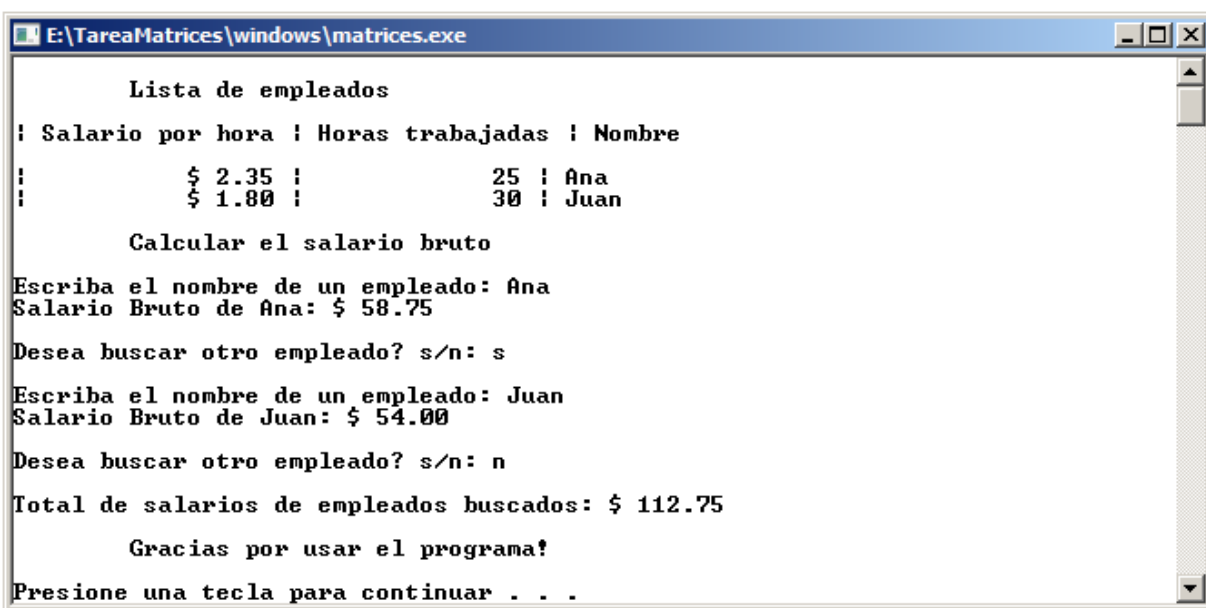
- Byron S. Gottfried, Programación en C. McGraw-Hill 1991.
- L. Joyanes, Estructura de Datos en C. McGraw-Hill 2007.
- C. Osvaldo, Estructura de Datos, Tercera Edición, McGraw-Hill 2007.

Estrategias de búsqueda en un arreglo: las estrategias de búsqueda que utilizamos en este proyecto consistieron en recorrer todo el arreglo utilizando los índices y realizar la comparación de caracteres con ayuda de la función *strcmp* de la librería string.

IDENTIFICACIÓN DE LA SALIDA

Para la realización del proyecto esperamos que el programa nos devuelva en la salida el nombre del empleado junto con su salario bruto. Luego nos imprime la cantidad total de dinero de la planilla de los empleados buscados anteriormente.

Para más facilidad por parte de los usuarios se enlistaron todos los empleados antes de hacer la primera búsqueda, así el usuario no tendrá dificultad en elegir y recordar el empleado para calcular su salario bruto. Observe la siguiente pantalla:



```
E:\TareaMatrices\windows\matrices.exe

Lista de empleados
! Salario por hora ! Horas trabajadas ! Nombre
!           $ 2.35 !           25 ! Ana
!           $ 1.80 !           30 ! Juan

Calcular el salario bruto
Escriba el nombre de un empleado: Ana
Salario Bruto de Ana: $ 58.75

Desea buscar otro empleado? s/n: s
Escriba el nombre de un empleado: Juan
Salario Bruto de Juan: $ 54.00

Desea buscar otro empleado? s/n: n
Total de salarios de empleados buscados: $ 112.75

Gracias por usar el programa!
Presione una tecla para continuar . . .
```

Figura 1. Presentación de lista de empleados y búsqueda.

Los demás detalles del programa se explican en la sección de evidencias.

ALGORITMO

Estructura de la lógica: nuestro proyecto consiste en dos matrices, una que almacena los empleados y otra que almacena en una fila los salarios por hora y en otra fila las horas trabajadas. Con esto calcularemos el salario bruto de los empleados solicitados. El proceso sería el siguiente:

```

// Programa principal, Repositorio de codigo: http://github.com/nodetino/tarea\_matrices
#include <stdio.h> // Librería de entrada y salida
#include <stdlib.h> // Librería que contiene la función system
#include <string.h> // Librería para manejar cadenas

#define MAX 5 // Límite de empleados

int main() {
    int n, i, existe; // Variables utilizadas n: cantidad de empleados, i: contador
    float salario[2][5], total_sal=0; // Matriz de salarios y horas trabajadas
    char empleados[MAX][20], empleado[20], otro='s'; // Arreglos de nombres de empleados
    printf("\n\tPrograma para la gestion de empleados\n\n"); // Solo admite el limite MAX
    do {
        printf("Escriba la cantidad de empleados de 1 a %d: ", MAX);
        scanf("%d", &n);
        if (n <= 0 || n > MAX) {
            printf("Error: El rango de empleados es de 1 a %d\n", MAX);
        }
    } while (n <= 0 || n > MAX);
    for (i=0; i < n; i++) { // Lee los nombres, salarios y horas trabajadas
        printf("\nEscriba el nombre del empleado No. %d: ", i+1);
        scanf("%s", empleados[i]);
        printf("Escriba el salario por hora de %s: ", empleados[i]);
        scanf("%f", &salario[0][i]);
        printf("Escriba las horas trabajadas de %s: ", empleados[i]);
        scanf("%f", &salario[1][i]);
    }
    system("cls"); // Limpia la pantalla
    printf("\n\tLista de empleados\n"); // Imprime la lista de empleados
    printf("\n| Salario por hora | Horas trabajadas | Nombre \n\n");
    for (i=0; i < n; i++) {
        printf("| $ %.2f | %.0f | %s \n", salario[0][i], salario[1][i], empleados[i]);
    }
    printf("\n\tCalcular el salario bruto\n");
    do { // Búsqueda de empleados y cálculo del salario bruto
        existe=0;
        printf("\nEscriba el nombre de un empleado: ");
        scanf("%s", empleado);
        for (i=0; i < n; i++) {
            if (strcmp(empleados[i], empleado) == 0) {
                printf("Salario Bruto de %s: $ %.2f\n", empleados[i], salario[0][i]*salario[1][i]);
                total_sal = total_sal + (salario[0][i]*salario[1][i]);
                existe=1;
            }
        }
        if (existe == 0) {
            printf("El empleado %s no existe\n", empleado);
        }
        printf("\nDesea buscar otro empleado? s/n: "); // Solicita si desea terminar
        scanf("%s", &otro);
    } while (otro != 'n');
    printf("\nTotal de salarios de empleados buscados: $ %.2f\n", total_sal); // Total
    printf("\n\tGracias por usar el programa!\n\n"); // Mensaje de agradecimiento
    system("pause");
    return 0;
}

```

Excepciones: en el proyecto realizado estamos trabajando con estructuras de datos estáticas, por lo tanto el límite debe ser declarado. Nuestro programa identifica si la cantidad de empleados que desea introducir es mayor a la permitida por el arreglo. Observe la siguiente pantalla:

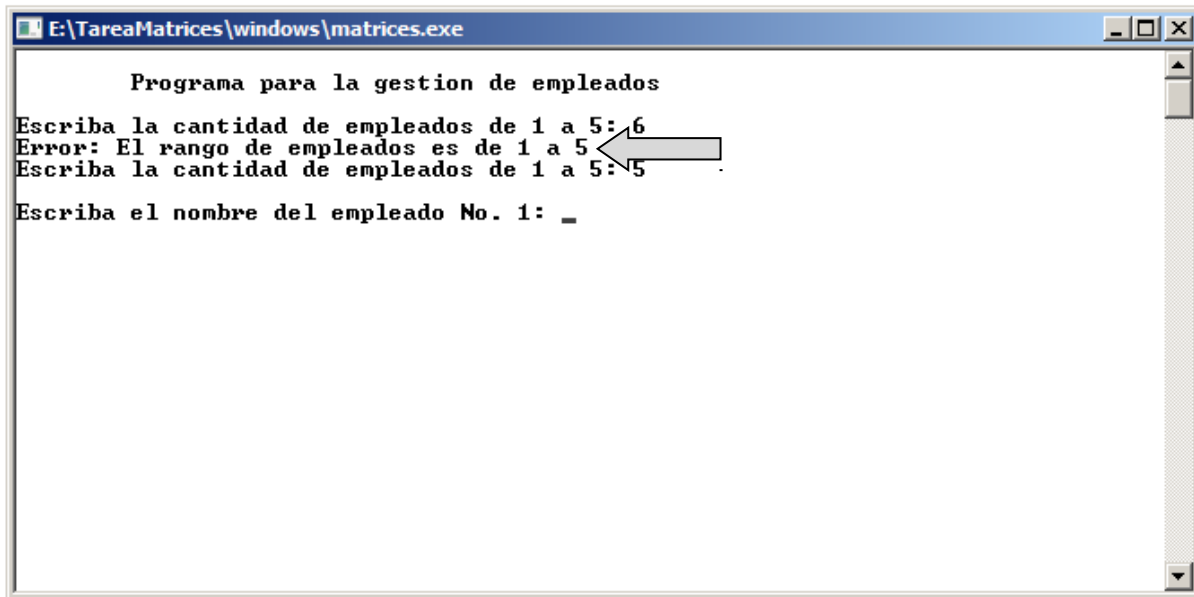


Figura 2. Presentación de error al determinar la cantidad de empleados.

Los demás detalles del programa se explican en la sección de evidencias.

Debemos aclarar que nuestro proyecto no maneja todos los errores, ya que si un usuario introduce un nombre con espacios, caracteres especiales no ASCII, una letra en la sección del salario o en las horas sucederá un error grave, *inutilizando el programa*. Esperemos que en el futuro podamos verificar si los datos introducidos por el usuario son los esperados, especialmente cuando pedimos el salario.

Eficiencia: en el proyecto hicimos el uso de ciclos para recorrer las estructuras de datos. Los ciclos más eficientes para recorrer los arreglos presentados fueron los *for* ya que teníamos un tamaño determinado. También hicimos uso de los ciclos *do* para preguntar la cantidad de empleados válida y además utilizamos la menor cantidad de variables posibles para ahorrar memoria. *Ver Cuadro 1.*

Evidencia: para el comprobar el funcionamiento del programa tenemos las siguientes pantallas:

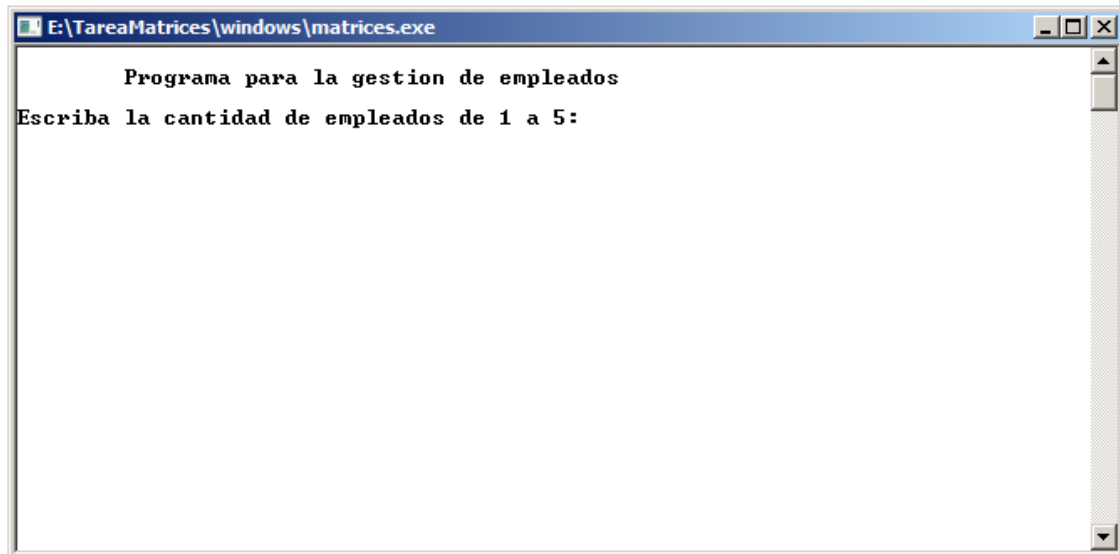


Figura 3.1. Presentación del programa. Primero se pide al usuario la cantidad de empleados.

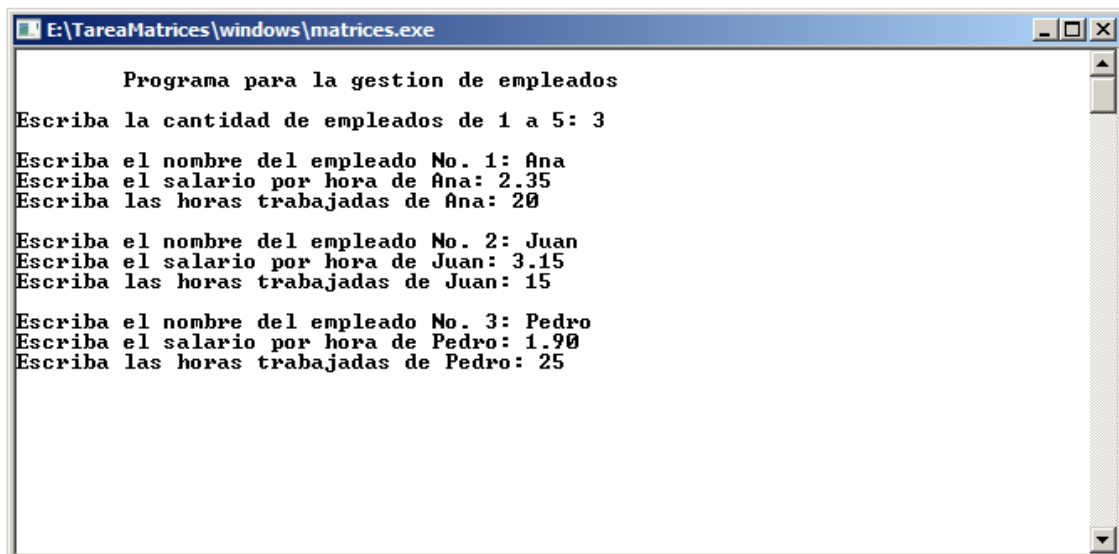


Figura 3.2. Luego se pide al usuario que introduzca todos los datos solicitados, el nombre del empleado, el salario por hora del empleado y sus horas trabajadas.


```
E:\TareaMatrices\windows\matrices.exe

Lista de empleados
! Salario por hora ! Horas trabajadas ! Nombre
!
!      $ 2.35 !      20 ! Ana
!      $ 3.15 !      15 ! Juan
!      $ 1.90 !      25 ! Pedro

Calcular el salario bruto
Escriba el nombre de un empleado: Julio
El empleado Julio no existe
Desea buscar otro empleado? s/n: s
Escriba el nombre de un empleado: Ana
Salario Bruto de Ana: $ 47.00
Desea buscar otro empleado? s/n: s
Escriba el nombre de un empleado: Pedro
Salario Bruto de Pedro: $ 47.50
Desea buscar otro empleado? s/n:
```

Figura 3.3. Presentación de la lista de los empleados introducidos para que al usuario le sea menos difícil recordar los nombres de los empleados. Luego se pide el nombre de un empleado y si existe se imprime el salario bruto del mismo.

```
E:\TareaMatrices\windows\matrices.exe

Lista de empleados
! Salario por hora ! Horas trabajadas ! Nombre
!
!      $ 2.35 !      20 ! Ana
!      $ 3.15 !      15 ! Juan
!      $ 1.90 !      25 ! Pedro

Calcular el salario bruto
Escriba el nombre de un empleado: Julio
El empleado Julio no existe
Desea buscar otro empleado? s/n: s
Escriba el nombre de un empleado: Ana
Salario Bruto de Ana: $ 47.00
Desea buscar otro empleado? s/n: s
Escriba el nombre de un empleado: Pedro
Salario Bruto de Pedro: $ 47.50
Desea buscar otro empleado? s/n: n
Total de salarios de empleados buscados: $ 94.50
Gracias por usar el programa!
Presione una tecla para continuar . . .
```

Figura 3.4. Finalmente de que el usuario pida salir del programa y se presenta la cantidad total dinero en salario de los empleados buscados.

DETERMINACIÓN DE LA ENTRADA

Estrategias de prueba del programa y recursos del lenguaje C necesarios: para probar el programa hemos utilizado valores dependiendo del tipo de dato que se requiera.

Para los empleados utilizamos nombres sin caracteres especiales; es decir sin *eñes* ni *tildes*, ya que en C no manejamos muy bien los caracteres no ASCII.

En los salarios y las horas trabajadas utilizamos números flotantes ya que estos generalmente ameritan el uso de decimales o fracciones de la unidad, por ejemplo: \$ 2.35 en el salario ó 20 horas y media en las horas trabajadas.

Para hacer las pruebas utilizamos valores pequeños, por cuestiones de comodidad, ya que si podemos hacer que el programa funcione para algunos pocos, este funcionará para muchos si ajustamos la cantidad.

Un detalle que tomamos en cuenta a la hora de realizar el programa fue la definición de una constante MAX que nos permite ajustar el límite máximo de empleados permitidos de una forma sencilla, ya que sin ella tendríamos que revisar por todo el programa para cambiar los valores y ajustar un nuevo límite de empleados

CONCLUSIONES

Al realizar este proyecto podemos concluir que:

1. Los arreglos son la estructura de datos más básica que podemos utilizar de manera estática.
2. Los arreglos pueden tener una o varias dimensiones, los vectores y matrices son ejemplos de ello.
3. Los índices y el número de elemento de un arreglo no son lo mismo. Los índices se comienzan a contar desde cero (0).
4. En un proyecto siempre hay que determinar cuál va a ser la salida, proceso y entrada.
5. La lógica del proyecto es el núcleo del mismo, ya que gracias a ella se permite el flujo de los datos a través del programa.
6. Un programa debe ser capaz de manejar errores e informarle al usuario sobre los mismos.
7. La usabilidad define si el usuario será capaz de utilizar sin ningún problema el programa.
8. Debemos tener claro que la investigación y el asesoramiento de profesionales son puntos clave en el correcto desarrollo del proyecto.
9. El lenguaje de programación C es un lenguaje estructurado de tipado fuerte que nos permite conocer a fondo la base de la computación.
10. La utilización de herramientas como los entornos de desarrollo integrado ayudan en la ardua tarea de escribir código, marcando errores de sintaxis y facilitando las tareas de compilación.

COEVALUACIÓN

Para realizar el proyecto tuvimos dificultades como la realización de búsquedas en los arreglos; pero estas debilidades fueron inmediatamente corregidas con la ayuda de libros y preguntas en clase.

Se obtuvieron nuevos conocimientos en el manejo de las funciones de librerías como `string.h`, nosotros logramos concretar el proyecto ayudándonos de la investigación y el desarrollo de preguntas que nos ayudaban a pensar mejor sobre el tema.

Como conocimientos previos, tuvimos que saber acerca de la sintaxis del lenguaje de programación C y el desarrollo de la lógica esencial para todo programador.

En este proyecto valoramos el trabajo de investigación que se realizó, todos trabajamos en grupo haciendo preguntas al profesor y entre nosotros mismos. También hacemos mención a lo importantes que son las herramientas de internet como lo es Git. En la realización de este proyecto mi compañero y yo, logramos utilizar esta herramienta para manejar las versiones de nuestro proyecto. Git tiene muchas ventajas ayudando en la localización de errores y en la estructuración del código. Plataformas como Github, permiten un uso más social de Git. En ella se puede publicar el código en repositorios que todo mundo puede ver si se desea. para este proyecto nosotros hemos almacenado todo el trabajo realizado en la siguiente dirección: http://github.com/nodetino/tarea_matrices

Si se desea se puede hacer una revisión del código, guardarlo en favoritos y hasta hacer mejoras en el mismo. Todo a través de Github.