Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE0117 – Programación Bajo Plataformas Abiertas
III ciclo 2022

Proyecto C

Vargas Romero Fernando B78114
Rojas Ocampo Jose Rodolfo B36093
Campos Castro Kevin B91519

Profesor: Julián Gairaud

05 de Marzo, 2023

1. Parte 1: Escaleras (36 %)

Una persona quiere subir una escalera de N escalones. Para hacerlo puede subir en pasos de 1, 2, 3, o 4 escalones. Por ejemplo, para una escalera de 4 escalones, la persona podría subir usando cualquiera de estas combinaciones:

- 4 pasos de 1 escalón.
- 1 paso de 1 escalón, más un paso de 3 escalones.
- 2 pasos de 2 escalones.
- 1 paso de 1 escalón, 1 paso de 2 escalones, y finalmente 1 paso de 1 escalón.
- 1 paso de 4 escalones.

(no se incluyen todas las opciones en este ejemplo).

Cree un programa que reciba un número entero, que representará la cantidad de escalones de una escalera "N".

El programa debe imprimir TODAS las combinaciones de pasos posibles para llegar al tope de la escalera. El número debe ingresarse mediante **LA LÍNEA DE COMANDOS** (ver ejemplo, que puede estar incompleto). Cada combinación debe ser impresa usando el formato del ejemplo: números separados por coma, donde la suma de todos los números debe ser N, y cada número debe estar en el rango de 1 a 4 (inclusive).

Puede compilarlo así: **$gcc proyecto1.c -o exe**.

Ejemplo:
$ ./exe 4
Las combinaciones para subir una escalera de 4 escalones son:
1,1,1,1
1,1,2
1,2,1
2,1,1
2,2
1,3
3,1
4

Aspectos a evaluar (los puntos suman 100, que equivalen al 36 %):

1. **(10 pts)** El programa debe recibir la cantidad de escalones por la **línea de comandos.**
2. **80 pts** El programa imprime exitosamente todas las combinaciones para cualquier número N.
3. **10 pts**El programa reconoce estos errores y da un mensaje de error en consola:
   - Más argumentos de la cuenta (sólo se puede recibir uno).
   - Menos argumentos de la cuenta.
   - Que N no sea un entero positivo (el caso de N=0 puede dar un error genérico, o bien imprimir una lista vacía)

La solución de este ejercicio será subida en MV1 bajo el nombre **proyecto1.c**

2. Parte 2: Ordenamiento descendente (36 %)

Cree un programa que reciba mediante **LA LÍNEA DE COMANDOS** la dirección **ABSOLUTA** de un archivo. Este archivo contiene números flotantes desordenados, uno por línea:
5
6.7
9
2
4.6
79

El programa debe leer el archivo, obtener los números, e imprimirlos en pantalla en orden descendente, por ejemplo: 79
9
6.7
5
4.6
2
Puede compilarlo así: **$gcc proyecto2.c -o exe**
Ejemplo:
$ ./exe /tmp/lista_numeros.txt

Notas:

1. El algoritmo de ordenamiento de números debe ser implementado por el estudiante. NO es permitido usar funciones de ninguna biblioteca. Puede investigar el algoritmo "bubble sort", un algoritmo de ordenamiento ineficiente (pero fácil de entender), o implementar uno propio.

2. Se debe investigar cómo manejar archivos (leer en este caso).

La solución de este ejercicio será subida en MV1 bajo el nombre **proyecto2.c**

3. Parte 3: Memoria Dinámica (28 %)

Responda de manera ordenada las siguientes preguntas (cada una vale 7 %).

1. Refiriéndose a memoria en los sistemas operativos ¿Cuál es la diferencia entre Heap y Stack? (no confunda con las estructuras de datos del mismo nombre).

Heap y Stack son dos regiones de memoria diferentes en los programas en C que el programador tiene a su disposición. La memoria Stack es la que permite almacenar argumentos y variables locales durante la ejecución de las funciones en las que están definidas, es decir, almacenar datos temporales, las funciones al ser "invocadas" crean un "frame" en la memoria Stack, y al terminar la ejecución de la función, las variables de dicha función son eliminadas automáticamente, esto para liberar espacio. La memoria Heap es una memoria dinámica, es la que permite almacenar variables adquiridas de las funciones como "malloc" o "calloc" y que dichas variables persistan durante la ejecución del programa, o hasta liberar el espacio de la memoria utilizando "free" o puede ocurrir un "memory leak", el cuál duraría mientras el proceso siga corriente, y se liberará el espacio de memoria cuando el sistema operativo termina de correr, es decir, la memoria Heap no es limitada, y crece o decrece según las necesidades del programa. En resumen, la memoria Heap es una memoria dinámica, mientras que la memoria Stack es una memoria estática [1].

2. Describa la diferencia entre una variable reservada estática y dinámicamente. Mencione ventajas y desventajas de ambas.

Una variable estática no cambia su valor, este sigue siendo el mismo en todas las llamadas dentro de la función que la contiene, mientras que una variable dinámica puede cambiar su valor, siempre que sea requerido, en cada llamada dentro de la función que la contiene. Una ventaja de la memoria estática es que su espacio en memoria es asignado desde el inicio del programa, lo que la vuelve una variable de acceso rápido, sin embargo, tiene una desventaja, y es que si dicha variable no está en uso, ocupa espacio de memoria. La ventaja de las variables dinámicas es que su tamaño en memoria es cambiable, según se requiera durante la ejecución del programa, ya que las variables dinámicas cambian su valor y tipo (formato) pueden ocasionar errores durante la ejecución sino se tiene contemplado estos cambios durante la programación [1][2].

3. Cree una muestra de código (en el documento pdf, no necesita adjuntar código .c) donde se haga lo siguiente:

   a) Crear una función que reciba una cantidad N de elementos, y un puntero de tipo double.

   b) La función debe reservar N elementos del tipo double, y asignarlo al puntero de entrada.

   c) La función retorna un número entero: 0 si no hay errores, -1 si hubo algún error.

```c
// se agregan las librerias
#include <stdlib.h>
#include <stdio.h>
#include <memory.h>
// se define la funcion que recibe una N cantidad de elementos
// y el puntero de tipo double
int reserva(int N, double* puntero)
{
    // se filtran numeros negativos, el programara retornara un -1
    if (N <= 0)
    {
        return -1;
    }

    double* arreglo = (double*) malloc(N * sizeof(double));

    if (arreglo == NULL)
    {
        return -1;
    }

    for (size_t i = 0; i < N; i++)
    {
        arreglo[i] = 0.0;
    }

    memcpy(puntero, arreglo, N * sizeof(double));
    free(arreglo);
    // si todo sale bien el programa retorna un 0
    return 0;

}
```

4. Investigue qué es el concepto de “union” en C. Describa la diferencia entre un “unión” y un “struct”.

La “union” es una palabra reservada en el lenguaje C usada para almacenar datos, similar al “struct” pero se diferencian en el comportamiento. La diferencia entre “union” y “struct”, es que la primera solo albrega uno de los miembros de dicha lista incluida en la declaración de tipo, se reserva un espacio correspondiente al miembro de mayor tamaño, todos estos miembros apuntarán a la misma dirección de memoria, mientras que para el segundo se asigna espacios de memoria separados para cada miembro [3][4][5].

4. Bibliografía

[1] Universidad Tecnica Federico Santa Maria (2010, Abr 08). 4. Memoria, Stack, String [Online]. Disponible: http://profesores.elo.utfsm.cl/ tarredondo/info/datos-algoritmos/ELO-320-Memoria.pdf

[2] Techie Delight (s.f). Variables estáticas y no estáticas en Java [Online]. Disponible: https://www.techiedelight.com/es/difference-between-static-and-non-static-variables-java/

[3] UGR (s.f). Capitulo 7: estructura de datos [Online]. Disponible: https://ccia.ugr.es/ jfv/ed1/c/cdrom/cap7/cap76.htm

[4] B. Pulido (s.f). Objetos compuestos en C: estructuras y uniones [Online]. Disponible: https://www.infor.uva.es/ belar/Ampliacion/Cursos

[5] J. Daymon (2018, Mar 27). Diferencia entre estructura y unión [Online]. Disponible: http://diferenciaentre.net/diferencia-entre-estructura-y-union/