Guion de prácticas 5

Memoria Dinámica y Valgrind

DECSAI

Metodología de la Programación

Grado en Ingeniería Informática

Prof. David A. Pelta

Introducción al guion

En este guion se pondrán en práctica los conceptos asociados a la gestión dinámica de la memoria. Para ello se utilizarán arrays, matrices y funciones. Luego se describe (en un documento separado) la herramienta Valgrind que permite analizar el uso correcto de la memoria dinámica.

1. Arrays Dinámicos y parámetros a `main`

La sucesión de Fibonacci es la siguiente sucesión infinita de números naturales: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, \ldots$ definida de la siguiente manera:

$$\begin{aligned} f_0 &= 0 \\ f_1 &= 1 \\ f_n &= f_{n-1} + f_{n-2} \end{aligned}$$

Se pide implementar un programa `fibonacci` que permita generar y mostrar los primeros N términos de la sucesión. Los valores se almacenarán en un array de enteros v de tamaño n cuyo espacio en memoria se gestionará de manera dinámica. Cuando v esté completo, se redimensionará para permitir almacenar $EXTRA$ valores adicionales. **¡ Recuerde liberar la memoria !**

Inicialmente, $n = 2$, `v[0] = 0` y `v[1] = 1`. Los valores N y $EXTRA$ (ambos enteros) son parámetros de la función `main` (vea el ejemplo en la Fig. 1). Utilice `valgrind` para comprobar que el uso de la memoria es correcto.

La ejecución de

```
valgrind --leak-check=full fibonacci 100 3
```

donde $N = 100$ y $EXTRA = 3$, debe informar que no hay errores.

Puede implementar todo lo necesario en un único fichero.

2. Módulo “lienzo” (Matriz dinámica)

En este ejercicio utilizaremos una matriz dinámica representada mediante un array 1D de punteros a arrays 1D (repase los apuntes de teoría). La matriz almacenará valores de tipo `char` y por tanto, podemos entender dicha matriz como un “lienzo” donde podremos pintar. La posición (0,0) representa el vértice superior izquierdo del lienzo. Se pide implementar un módulo `lienzo` con las siguientes funciones (todas reciben como parámetro, al menos, una matriz M y la cantidad de filas y columnas nf, nc):

1. `reservaMemoria`: reserva memoria para M.
2. `liberaMemoria`: libera la memoria de ocupada por M.

3. `rellenar`: rellena M con el carácter (valor de tipo char) que se reciba como parámetro.

4. `imprimir`: muestra el contenido de M.

5. `dibujaRectangulo`: dada una posición (i, j) que representa el extremo superior izquierdo de un rectángulo, un valor de $ancho$ y otro de $alto$, la función "dibuja" en M el rectángulo indicado usando un carácter c que se recibe como parámetro. Si parte del rectángulo "cae" fuera de la matriz, esa parte se ignora. Si la posición (i, j) contiene valores negativos, a la/s coordenada/s incorrecta/s se le/s asigna el valor cero. Si se intenta dibujar en una matriz de tamaño 0, no se hace nada.

6. `combinar`: dadas dos matrices A, B con tamaños nf^A, nc^A, nf^B, nc^B, construye una nueva matriz C con las siguientes características:

   - $nf^C = max(nf^A, nf^B)$ y $nc^C = nc^A + nc^B$. En otras palabras, en C, la matriz B se "agrega" a la derecha de A.
   - Rellene primero la matriz C con el símbolo '.'. La matriz con menos filas, se copia en C alineada en la parte inferior.

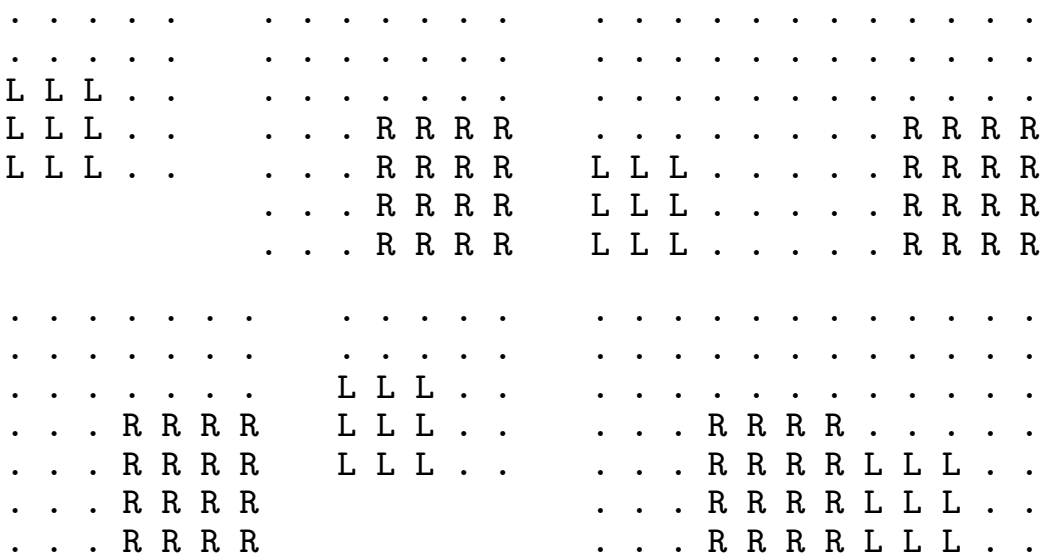
Ejemplos de `dibujaRectangulo`

Suponga que dispone de una matriz M, con $nf = nc = 5$ y se quiere dibujar un rectángulo de `ancho = lado = 3`. A continuación se muestra como quedaría la matriz utilizando distintos valores para las posiciones $(i, j) = (1, 1), (3, 3), (-1, -1)$

```
. . . . .     . . . . .     X X X . .
. X X X .     . . . . .     X X X . .
. X X X .     . . . . .     X X X . .
. X X X .     . . . X X     . . . . .
. . . . .     . . . X X     . . . . .
```

Ejemplos de `combinar`

```
. . . .     . . . .       . . . . . . . .
. A A .     . B B .       . . . . . B B .
. A A .     . B B .       . . . . . B B .
. A A .     . B B .       . A A . . B B .
. A A .     . B B .       . A A . . B B .
            . B B .       . A A . . B B .
            . B B .       . A A . . B B .
```



3. Tareas a realizar

3.1. Módulo `lienzo`

Implemente el módulo `lienzo` a partir del fichero `lienzo.h` y pruebe su implementación utilizando el fichero `main.cpp`. Ambos están disponibles en PRADO. Utilizando `valgrind`, compruebe que la gestión de memoria no contiene errores.

3.2. Gráfico de barras

Se pide implementar todo lo necesario para representar una serie de valores enteros en un gráfico de barras (utilizando las funcionalidades del módulo `lienzo`). Debe implementar dos funciones.
1) `void pintaBarras(char ** & G, int & nf, int & nc, int *datos, int n, char c)`: recibe una matriz, su tamaño, los datos a pintar (un array de enteros) y su tamaño, y el carácter para pintarlos. La función redimensionará G para acomodar los datos a pintar. Para construir el gráfico, se deben seguir los siguientes pasos:

1. Calcular tamaño del gráfico y reservar memoria. Para calcular el ancho, se considera que cada barra tendrá un ancho de 2 unidades, y estarán separadas entre sí por dos espacios. Para ello, se añade un espacio al comienzo y al final de cada barra. La altura vendrá determinada por el valor máximo de `datos` + 1.

2. Determinar las "coordenadas" de inicio de cada barra asociada a cada valor de `datos` y dibujarla con la función `dibujaRectangulo`. Recuerde que la posición (0,0) corresponde al vértice superior izquierdo.

Veamos algunos ejemplos del resultado esperado (se utiliza el símbolo '.' para representar los espacios)

`int v[3] = {3, 1, 2}`

```
............
.xx.........
.xx......xx.
.xx..xx..xx.
```

`int v[1] = {4}`

```
....
.xx.
.xx.
.xx.
.xx.
```

2) `void aniadeBarra(char ** & G, int & nf, int & nc, int dato, char c)`: recibe una matriz, su tamaño, un dato a pintar, y el carácter para pintarlo. La función redimensionará G para agregar un nuevo elemento. Para construir el nuevo gráfico, se deben seguir los siguientes pasos:

1. Crear una matriz auxiliar A con `dato+1` filas y 4 columnas. Pintar la barra correspondiente al valor `dato`.
2. '"Unir" las matrices G y A utilizando la función `combinar`.
3. Reasignar variables y liberar memoria.

```
#include <iostream>
#include <string>
#include <cstdlib>
using namespace std;

int main(int argc, char *argv[]){
/* int argc: número de argumentos usados al ejecutar el programa
   .
char *argv[]: array de C strings con cada uno de los argumentos.
argv[0]: nombre del ejecutable
argv[1]: primer argumento
...  */

int v1, v2;

cout << "Nombre del programa " << argv[0] << endl;
cout << "Parametros ";
for(int i = 0; i < argc; i++){
  cout << argv[i] << endl;
}

cout << "Paso de const char* a enteros (funcion atoi) " << endl;
v1 = atoi(argv[1]);
v2 = atoi(argv[2]);

cout << "Suma de v1+v2: " << v1 + v2 << endl;
}
```

Figura 1: Paso de parámetros a `main`