Laboratorio de Programación - Labo06

Taller de programación sobre matrices y tableros

Algoritmos y Estructuras de Datos I

Departamento de Computación, FCEyN, Universidad de Buenos Aires.

¿Qué es una matriz?

- Una matriz, en nuestro contexto, es simplemente un vector de dos dimensiones que tiene el mismo largo en cada uno de sus elementos.
- Para declarar una matriz (de enteros) pueden hacer:

```
1 vector<vector<int>> m;
```

En vez de *int* pueden poner cualquier otro tipo (*string*, *char*, etc).

¿Y qué hacemos con la matriz?

Muchas veces queremos utilizar matrices para representar estructuras como, por ejemplo:

- Matrices de verdad (esas de Álgebra que tienen determinante y esas cosas).
- Tableros (de ajedrez, por ejemplo).
- Mapas (por ejemplo, en cada casillero guardamos la altura del territorio en esas coordenadas, o la cantidad de personas que viven en una determinada manzana).
- Imágenes.
- Series temporales.
- Muchísimos etcéteras.

Operaciones (que vamos a necesitar) sobre matrices

- Declarar una matriz.

```
1 vector<vector<int>> m;
```

- Inicializar una matriz de m filas $\times$ n columnas con ceros.

```
1 vector<vector<int>> res(m, vector<int>(n));
```

- Inicializar una matriz de m filas $\times$ n columnas todas con el mismo valor (x).

```
1 vector<vector<int>> res(m, vector<int>(n, x));
```

- Inicializar una matriz con valores fijos,

```
1     vector<vector<int> > mat = {
2             {6, 12, 18},
3             {7, 14, 21},
4             {8, 16, 24},
5             {9, 18, 27}
6     };
```

Operaciones (que vamos a necesitar) sobre matrices

- Agregar una fila.

```
1 vector<vector<int>> m;
2 vector<int> v = {1,2,3};
3 m.push_back(v);
```

- Acceder a un elemento en la posición (i, j).

```
1 m[i][j]
```

Rotación de Matrices

Dada una matriz `mat` de $n \times m$ y dos enteros `d` y `a` queremos devolver una matriz con las m columnas movidas `d` veces a la derecha y las n filas movidas `a` veces hacia abajo.

Rotación de Matrices

Resolvamos el siguiente problema:

proc `rotar` (in mat: $seq\langle seq\langle \mathbb{Z} \rangle\rangle$, in d: $\mathbb{Z}$, in a: $\mathbb{Z}$, out res: $seq\langle seq\langle \mathbb{Z} \rangle\rangle$) {
  Pre $\{|mat| > 0 \wedge (\forall i : \mathbb{Z})(0 \leq i < |mat| \rightarrow_L |mat[i]| = |mat[0]|\}$
  Post $\{mismasDimensiones(res, mat) \wedge_L$
    $esLaMovidaParaAbajoYDerecha(res, mat)\}$
}

pred `mismasDimensiones` (m1: $seq\langle seq\langle \mathbb{Z} \rangle\rangle$, m2: $seq\langle seq\langle \mathbb{Z} \rangle\rangle$) {
  $|m1| = |m2| \wedge_L (\forall i : \mathbb{Z})\ (0 \leq i < |m1| \rightarrow_L |m1[i]| = |m2[i]|)$
}

pred `esLaMovidaParaAbajoYDerecha` (res: $seq\langle seq\langle \mathbb{Z} \rangle\rangle$, mat: $seq\langle seq\langle \mathbb{Z} \rangle\rangle$)
{
  $(\forall i, j : \mathbb{Z})\ (0 \leq i < |mat| \wedge_L 0 \leq j < |mat[i]| \longrightarrow_L res[i][j] = mat[(i - a)$
  mód $|mat|][(j - d)$ mód $|mat[i]|])
}

y donde mód es la operación módulo.

Rotación de Matrices

```
1  vector<vector<int>> rotar(vector<vector<int> > mat,
2                  int a, int d) {
3      int n = mat.size();
4      int m = mat[0].size();
5      vector<vector<int>> res(n, vector<int>(m));
6      int i = 0;
7      while(i < n) {
8          int j = 0;
9          while(j < m) {
10             res[i][j] = mat[(i - a) % n][(j - d) % m];
11             j++;
12         }
13         i++;
14     }
15     return res;
16 }
```

Matrices y más matrices

Durante la carrera verán más ejercicios de matrices hasta el cansancio en:

- Organización del Computador 2: Verán como aplicar filtro a imágenes (como los de Instagram) pero en lenguaje ASM.
- Algoritmos y Estructuras de Datos 3: Ejercicios sobre grafos, programación dinámica, etc.
- Métodos Numéricos: mejor conocida como "Matrices: la materia" (verán algoritmos sobre matrices como las de Álgebra).

Taller de Matrices

El taller de hoy tiene un enunciado y un archivo comprimido, como veníamos trabajando anteriormente. Dentro del archivo que se que se descarguen desde el folder del laboratorio de hoy, van a encontrar los siguientes archivos:

- `main.cpp`: Punto de entrada del programa.
- `ejercicios.cpp`: Aquí es donde van a volcar sus implementaciones.
- `ejercicios.h`: *headers* de las funciones que tienen que implementar.
- `casos.cpp`: Con una serie de diferentes pruebas a los ejercicios del taller.
- `casos.h`: Los *headers* de las funciones para poder llamarlas desde el `main.cpp`.