

ESTRUCTURAS DINÁMICAS DE DATOS

Profesor/es: SASIN Alejandro, CAÑETE Diego

Fecha de entrega: 14 / 06 / 2020

GRUPO N° 2

Nombre y Apellido	DNI
Borsani Franco	42043432
Casais Dario Fabian	42366715
Fiordalisi Santiago	35632506
Hernández Walter	38165627
Romero Ortiz Nicolás	40136661

2020



ALGORITMOS Y ESTRUCTURAS DE DATOS

Trabajo Práctico de Diagnóstico

1. Recomendaciones:

Al iniciar el desarrollo del trabajo práctico les pedimos que se cumplan con los límites establecidos dentro del enunciado, no traten de abarcar más de lo pedido aquí y, comiencen con racionalizar y comprender los problemas establecidos (pueden hacer un diagrama inicial en papel y lápiz o lo que sea necesario para interiorizar el enunciado).

2. Enunciado del Desarrollo:

Una empresa que produce artículos plásticos cuenta con una máquina que fabrica figuras (paralelepípedos) en diferentes colores y formas.

Para producir los artículos, la máquina que los fabrica cuenta con inyectores de diferentes colores (el número varía de 1 a 10), aunque no se sabe la cantidad de colores que vendrán en cada orden de producción. En cada partida se recibe la orden de producción en un archivo txt, donde se le dice la figura a producir, el color y los distintos parámetros a definir por el programador de la misma (radio, altura, etc.).

Una vez que termina un lote de producción (cada archivo es un lote), se necesita saber para cada figura:

- Que **Tipo de Figura** es.
- Que **Color** posee.
- Que **Cantidad de Plástico** se usó (sería la suma de las áreas de las figuras).
- Además, se necesitan saber:
 - Los **Totales** por Figura y por Figura-Color.
 - Dentro de cada color, mostrar en qué **Orden** se fueron produciendo las distintas figuras.

Algoritmos y estructuras de datos

- Un listado que muestre las figuras ordenadas por el tamaño de sus áreas.

Las figuras que produce la empresa son: **Círculos, Cilindros, Cubos, Triángulos y Rectángulos**. En el caso de los círculos en el archivo vendrá una línea que informará:

- Círculo (5,2):
 - Donde el valor numérico (5,2) corresponde al radio.
- Cilindro (3,2;4,3):
 - Donde el primer valor numérico (3,2) corresponde al radio y el segundo valor numérico (4,3) corresponde a la altura del cilindro.
- Cubo (2,5):
 - Donde el valor numérico corresponde a la longitud del lado.
- Triángulos (4; 2):
 - Los triángulos siempre son rectángulos y los valores informados corresponden al cateto mayor y al menor.
- Rectángulo (3; 4,5):
 - Donde el primer valor es la base y el segundo la altura.

Todos los valores son en cm. Como ejemplo se adjunta un archivo (FigurasEjemplo.txt) para las pruebas con el siguiente contenido:

```
círculo rojo 2,5
cubo azul 4
cilindro amarillo 2,5; 3
rectángulo amarillo 6; 3
triángulo azul 5; 2,5
círculo rojo 3
círculo azul 3,2
rectángulo rojo 1,4; 2
cilindro azul 3; 4
círculo amarillo 2,5
cubo azul 4
cilindro Rojo 2,5; 3
rectángulo azul 3; 5
triángulo rojo 3; 1,5
círculo amarillo 3,2
rectángulo azul 1,4; 2
cilindro rojo 3; 4
círculo azul 2,5
```

Algoritmos y estructuras de datos

cubo amarillo 4
cilindro rojo 2,5; 3
rectángulo azul 3,5
triángulo amarillo 4; 2
círculo azul 3
círculo rojo 3,2
rectángulo rojo 1,4; 2
cilindro azul 3; 4

3. DESARROLLO, IMPLEMENTACIÓN y FUNCIONAMIENTO:

Planteamiento del problema:

El proyecto fue implementado con el uso de listas simplemente enlazadas para las listas de las figuras, donde cada nodo contiene los datos correspondientes (una figura) y un puntero a un nodo siguiente del mismo tipo. Respecto a cada figura, las mismas son estructuras que contienen sus respectivos datos (color y dimensiones).

Fue lo más conveniente el uso de listas lineales debido a que la utilización de una pila no hubiese sido viable por el hecho de que un punto pide mostrar figuras por color y orden de llegada, y las pilas al ser LIFO (Last In, First Out), hubiese dificultado el desarrollo. Por otra parte, se podrían haber utilizado colas (FIFO) para optimizar recursos, pero elegimos listas ya que éstas nos permiten ingresar elementos en cualquier lugar y orden dentro de las mismas, permitiendo mostrar la impresión de menor a mayor de acuerdo al área.

Para desarrollar el proyecto, lo que se hizo fue lo siguiente: se tienen estructuras con cada tipo de figura leída (círculo, cilindro, triángulo, rectángulo, cubo) y listas para cada tipo.

Al comenzar, se instancian todas posibles figuras y las listas (vacías) y se abre el archivo de texto en modo lectura con el fin de leer los datos que contiene (si por algún motivo falla, se mostrará un mensaje de error y se saldrá de la ejecución).

Primeramente, se lee la figura de la que se trata y el color. Posterior a esto, se verifica si el color leído ya había sido cargado en algún momento dentro de una lista de colores (estructura) que contiene a los mismos, mediante la función **buscarColor()**:

La misma recorre la lista de colores para actualizarla con los colores que se fueron utilizando durante la impresión de los objetos del lote (archivo), si ingresa un color nuevo lo agrega.

```
94  bool buscarColor(NodoColor *inicio, string color){
95      bool b =false;
96      while (b==false && inicio != nullptr ){
97          if (inicio->data == color){
98              b=true;
99          }
100         inicio = inicio->sgte;
101     }
102     return b;
103 }
```

Si la carga ha superado los diez colores, se informa al usuario y se cierra el programa.

Una vez establecido el color, se valida el tipo de figura. En caso de ser un cilindro, un rectángulo o un triángulo, lo que se hace es separar los datos de las dimensiones, debido a que estas figuras tienen dos parámetros, a diferencia de las otras 2, y se guardan por separado (dato 1 y dato 2); continuo a esto se llama a la función **cambiarComaPorPunto()** que realiza la conversión del string que se lee del archivo txt reemplazando la coma por un punto (para trabajar con la notación adecuada).

```

70  string cambiarComaPorPunto(string dato){
71      for(int i=0; i<dato.size(); i++){
72          if(dato[i]==','){
73              dato[i]='.';
74          }
75      }
76      return dato;
77  }

```

Ahora, teniendo el color, la figura y las dimensiones ya se puede pasar a la creación, pero antes de esto se realiza un paso extra:

Se dispone de un Enum con el tipo de figuras que “soporta” el programa, luego con la función **convertirTipo()** realizamos la conversión de tipo string (que es como viene del archivo de texto leído) a enum para utilizar su valor en el switch e ir cargando las listas y creando las figuras. Si hay alguna figura que no corresponde con las soportadas devuelve otro para mostrar un mensaje de error con el tipo de objeto que se quiso imprimir.

```

22  //TODAS LAS FIGURAS QUE PODRÍAN SER CARGADAS.
23  enum tipoFigura {
24      circulo,
25      cilindro,
26      cubo,
27      triangulo,
28      rectangulo,
29      otro
30  };
46  tipoFigura convertirTipo(const string &figura) {
47      if (figura == "circulo")    return circulo;
48      if (figura == "cilindro")   return cilindro;
49      if (figura == "cubo")       return cubo;
50      if (figura == "triangulo")  return triangulo;
51      if (figura == "rectangulo") return rectangulo;
52      return otro;
53  }
54

```

Dependiendo del tipo de figura que se retorne, se utilizará una validación para crear la respectiva figura y ser añadida a su respectiva lista. Además, se crea la estructura “figura” y se añade a la lista de figuras tal como se explica ([VER](#)).

Los headers contienen los axiomas, pre y post condiciones necesarias para que el programa funcione correctamente. Están definidas las estructuras que vamos a utilizar y las primitivas a implementar, las cuales tienen también Pre y Post condiciones, los parámetros que reciben, el tipo de retorno y una breve descripción de la funcionalidad del caso de uso.

```

1  #ifndef CUBO_H_INCLUDED
2  #define CUBO_H_INCLUDED
3  #include <iostream>
4
5  using namespace std;
6
7  /*
8   Definicion del Tipo de dato para el manejo de Cubos
9   Atributos:
10    color
11    lado
12
13   Axiomas:
14    color != empty
15    lado > 0
16  */
17
18  //Definiciones de Tipos de Datos
19  typedef struct{
20    string color;
21    float lado;
22  }Cubo;
23
24
25  /*-----Definiciones de primitivas-----*/
26  /*
27  Descripción del caso de uso: al cubo creado desde el main se le asigna el color y la longitud de un lado.
28
29  Condiciones:
30    PRE: El cubo no debe haber sido creado.
31    POST: El cubo queda creado para su posterior uso.
32
33  Parametros:
34    referencia del cubo (instancia sobre la cual se invoca la primitiva); color y longitud del lado que le serán asignados.
35
36  Retorno: procedimiento (sin retorno).
37  */
38  void crearCubo(Cubo &cubo, string color, float lado);
39  /*-----*/

```

En los .cpp de cada figura implementamos las funciones primitivas definidas en

```

1  #include "cubo.h"
2  using namespace std;
3
4  //-----IMPLEMENTACION DE PRIMITIVAS-----//
5
6  void crearCubo(Cubo &cubo, string color, float lado){
7    cubo.color = color;
8    cubo.lado = lado;
9  }
10
11  //-----//
12  void setColor (Cubo &cubo, string color){
13    cubo.color = color;
14  }
15  //-----//
16  string getColor (Cubo &cubo){
17    return cubo.color;
18  }
19  //-----//
20  void setLado (Cubo &cubo, float lado){
21    cubo.lado = lado;
22  }
23  //-----//
24  float getLado (Cubo &cubo){
25    return cubo.lado;
26  }
27  //-----//
28  float calcularArea (Cubo &cubo){
29    return ( 6 * cubo.lado * cubo.lado );
30  }

```


el header correspondiente a cada una de las figuras.

Por cada figura hay una lista en la cual se agregan los objetos a medida que van ingresando. La función **agregarNodo()** es la que se encarga de actualizar la lista con el objeto nuevo y **lo agrega en orden** según el tamaño de su área.

```

8  NodoCubo* agregarNodo (NodoCubo* inicio, Cubo cubo){
9      NodoCubo* nuevo = new NodoCubo;
10     nuevo->data = cubo;
11     nuevo->sgte = nullptr;
12     if(inicio == nullptr || calcularArea(nuevo->data) < calcularArea (inicio->data)){
13         nuevo->sgte = inicio;
14         inicio = nuevo;
15     }
16     else{
17         NodoCubo* aux = inicio;
18         //evaluo del segundo en adelante
19         while(aux->sgte!=nullptr && calcularArea(aux->sgte->data) < calcularArea(nuevo->data)){
20             aux = aux->sgte;
21         }
22         if(aux->sgte!=nullptr){
23             nuevo->sgte = aux->sgte;
24         }
25         aux->sgte = nuevo;
26     }
27     return inicio;
28 }

```

Se procede a cerrar el archivo y se inicializa el menú que verá el usuario, el cual cuenta con 4 opciones:

- Figuras por tamaño de áreas junto con plástico utilizado por el total de un tipo.
 - Cantidad de plástico por figura y color.
 - Orden de llegada por figura – color.
 - Finalizar.
- a) Se envía la lista de la figura que se desea imprimir, la cual se recorre nodo a nodo mostrando por cada uno el color, el / los datos ingresados y el área (llamando a la función **calcularArea()**), para finalmente mostrar la cantidad total de plástico.

```

30 void imprimir(NodoCubo *inicio){
31     double total = 0;
32     for (NodoCubo* p = inicio; p!=nullptr; p=p->sgte){
33         cout<<"Cubo " << "color: "<< getColor(p->data)<< " lado: "<< getLado (p->data)<< " area: "<< calcularArea(p->data)<<" " <<endl;
34         total = total +calcularArea(p->data);
35     }
36     cout<<"La cantidad total de plastico de los Cubos es: " << total <<endl;
37 }

```

- Con la función **imprimirPorColor()**, se recorre cada nodo de las listas y se hace una separación por color, mostrando el consumo de plástico por color y figura.


```

39 void imprimirPorColor(NodoCubo *inicio, NodoColor *inicioColor){
40     for (NodoColor* c = inicioColor; c!=nullptr; c=c->sgte){
41         double total = 0;
42         for (NodoCubo* p = inicio; p!=nullptr; p=p->sgte){
43             if(getColor(p->data)==c->data){
44                 total = total + calcularArea(p->data);
45             }
46         }
47         cout<<"La cantidad total de plastico de los cubos de color "<< c->data << " es: " << total << endl;
48     }
49 }

```

- c) Llamando al método **imprimirOrdenColor()**, se recorre la lista de “figuras” y con la lista de color se clasifica por orden de llegada.

Por último, cuando se sale de la ejecución (opción cero), se libera la memoria consumida por las diferentes listas.

DATOS A DESTACAR:

Algunas librerías importantes utilizadas son `fstream` (la cual define clases para operaciones de lectura, escritura y lectura / escritura en archivos) y `string.h` (para operaciones con cadenas de caracteres, como `string compare`).

4. CASOS DE USO QUE REQUIEREN EXPLICACIÓN:

Además de tener un .cpp y un .h de cada una de las figuras y de las listas, se tiene un figura.cpp y un figura.h junto con su respectiva lista. Ésta estructura se crea una vez que se "construye" la figura introducida por teclado y se agrega a su respectiva lista:

```
crearFigura(fig, figura, color, "("+dato1+"");
listaFigura = agregarNodo(listaFigura, fig);
```

Lo
s
parámetr

os que recibe el primer método son una estructura figura (fig), un nombre (figura), un color

(color)

y los

valore

s

leídos

del

archivo (dato 1, y dato 2 en caso de ser necesario).

```
void crearFigura(Figura &figura, string nombre, string color, string atributos){
    figura.nombre = nombre;
    figura.color = color;
    figura.atributos = atributos;
}
```

Posteriormente, la figura se añade a la "listaFigura", que contendrá todas las figuras leídas del teclado (figura y puntero nodo a la siguiente figura).

La utilidad de "Figura" es utilizada para mostrar por pantalla, con cada uno de los colores disponibles, las figuras por orden de llegada, teniendo un método (imprimirOrdenColor) que recibe la lista de todas las figuras y la lista de todos los colores.

```
void imprimirOrdenColor(NodoFigura *inicio, NodoColor *inicioColor){
    for (NodoColor* c = inicioColor; c!=nullptr; c=c->sgte){
        int num = 1;
        cout<<"Color " << c->data << " por orden de llegada: " <<endl;
        for (NodoFigura* p = inicio; p!=nullptr; p=p->sgte){
            if (getColor(p->data)== c->data){
                cout<< num <<"- " << getNombre(p->data) << " " << getAtributo(p->data) <<endl;
                num++;
            }
        }
        cout<<endl;
    }
}
```

RESULTADOS DE LA FUNCIÓN:

```
Color rojo por orden de llegada:
1- circulo (2.5)
2- circulo (3)
3- rectangulo (1.4; 2)
4- cilindro (2.5; 3)
5- triangulo (3; 1.5)
6- cilindro (3; 4)
7- cilindro (2.5; 3)
8- circulo (3.2)
9- rectangulo (1.4; 2)
```

5. Manual del Usuario

1) Formato del archivo: .txt

Nombre: FigurasEjemplo.txt

Formato de los datos: figura color dato1; dato2

El usuario deberá crear el archivo txt agregando los respectivos datos y cumpliendo el orden solicitado.

Dato 1: Depende de la figura

- En el caso de los círculos y cilindros corresponde al radio
- En el caso del cubo corresponde a la longitud del lado
- Considerando que los triángulos sean rectángulos, corresponde al cateto mayor y al cateto menor
- En el caso de los rectángulos corresponde a la base

Dato 2: Este dato aparece solo en los cilindros, rectángulos y triángulos.

Tanto en el caso del Dato 1 como en el Dato 2 los valores están expresados en centímetros, el uso del “;” es para mostrar la división entre los dos números.

Ejemplo: Cilindro rojo 2,5; 5

2) Uso del Programa

Al momento de correr el programa, nos aparecerá el siguiente menú:

1. Lista de figuras por tamaño de sus áreas y cuánto plástico usa
2. Lista de cantidad de plástico utilizado por figura y color
3. Lista en función del color y el orden en el que se cargaron las distintas figuras
4. Finalizar programa

La forma de acceder a cada opción es ingresar el número correspondiente a cada opción en el menú.

Si se desea salir del programa se debe volver al menú y usar la opción “Finalizar programa”.

Los datos se mostrarán de la siguiente forma, dependiendo la opción que se usa:

1. Lista de figuras por tamaño de sus áreas y cuanto plástico usa

Algoritmos y estructuras de datos

```
Circulo color: rojo radio: 2.5 area: 19.6349
Circulo color: rosado radio: 2.5 area: 19.6349
Circulo color: negro radio: 3 area: 28.2743
Circulo color: gris radio: 3.2 area: 32.1699
La cantidad total de plastico de los circulos es: 99.7141

Cilindro color: amarillo radio: 2.5 altura: 3 area: 86.3937
Cilindro color: violeta radio: 3 altura: 4 area: 131.947
La cantidad total de plastico de los cilindros es: 218.34

Triangulo color: marron cateto mayor: 5 cateto menor: 2.5 area: 6.25
La cantidad total de plastico de los triangulos es: 6.25

Cubo color: azul lado: 4 area: 96
La cantidad total de plastico de los Cubos es: 96

Rectangulo color: naranja base: 1.4 altura: 2 area: 2.8
Rectangulo color: verde base: 6 altura: 3 area: 18
La cantidad total de plastico de los rectangulos es: 20.8
```

2. Lista de cantidad de plástico utilizado por figura y color

```
La cantidad total de plastico de los cilindros de color rojo es: 0
La cantidad total de plastico de los cilindros de color azul es: 0
La cantidad total de plastico de los cilindros de color amarillo es: 86.3937
La cantidad total de plastico de los cilindros de color verde es: 0
La cantidad total de plastico de los cilindros de color marron es: 0
La cantidad total de plastico de los cilindros de color negro es: 0
La cantidad total de plastico de los cilindros de color gris es: 0
La cantidad total de plastico de los cilindros de color naranja es: 0
La cantidad total de plastico de los cilindros de color violeta es: 131.947
La cantidad total de plastico de los cilindros de color rosado es: 0
```

Algoritmos y estructuras de datos

3. Lista en función del color y el orden en el que se cargaron las distintas figuras

```
Color rojo por orden de llegada:
1- circulo (2.5)

Color azul por orden de llegada:
1- cubo (4)

Color amarillo por orden de llegada:
1- cilindro (2.5; 3)

Color verde por orden de llegada:
1- rectangulo (6; 3)

Color marron por orden de llegada:
1- triangulo (5; 2.5)

Color negro por orden de llegada:
1- circulo (3)

Color gris por orden de llegada:
1- circulo (3.2)

Color naranja por orden de llegada:
1- rectangulo (1.4; 2)

Color violeta por orden de llegada:
1- cilindro (3; 4)
```

6. Casos de prueba con resultados esperados:

En este ejemplo se introducen más de 10 colores diferentes, al suceder esto el programa tira un mensaje de error notificando el problema y se cierra la ejecución.

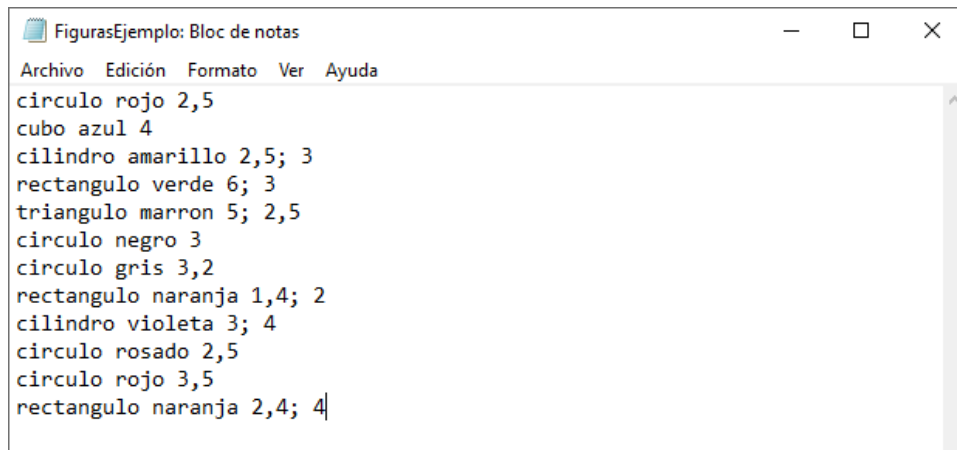
```

Archivo Edición Formato Ver Ayuda
circulo rojo 2,5
cubo azul 4
cilindro amarillo 2,5; 3
rectangulo amarillo 6; 3
triangulo azul 5; 2,5
circulo rojo 3
circulo azul 3,2
rectangulo rojo 1,4; 2
cilindro azul 3; 4
circulo amarillo 2,5
cubo azul 4
cilindro rojo 2,5; 3
rectangulo azul 3; 5
triangulo rojo 3; 1,5
circulo amarillo 3,2
rectangulo azul 1,4; 2
cilindro rojo 3; 4
circulo azul 2,5
cubo amarillo 4
cilindro rojo 2,5; 3
rectangulo azul 3,5; 6
triangulo amarillo 4; 2
circulo azul 3
circulo rojo 3,2
rectangulo rojo 1,4; 2
cilindro azul 3; 4
triangulo verde 4; 2
circulo rosado 3
circulo fuxia 3,2
rectangulo marron 1,4; 2
cilindro celeste 3; 4
circulo lila 2,5
cubo negro 4
cilindro naranja 2,5; 3
rectangulo gris 6; 3
  
```

```

C:\Users\Naryadg\Desktop\lectura\bin\Debug\lectura.exe
Hay mas de 10 colores. Verifique el archivo de carga.
Presione una tecla para continuar . . .
  
```


En este otro ejemplo se utilizan 10 colores simultáneamente para demostrar el máximo de colores permitidos por el programa:



```
FigurasEjemplo: Bloc de notas
Archivo Edición Formato Ver Ayuda
circulo rojo 2,5
cubo azul 4
cilindro amarillo 2,5; 3
rectangulo verde 6; 3
triangulo marron 5; 2,5
circulo negro 3
circulo gris 3,2
rectangulo naranja 1,4; 2
cilindro violeta 3; 4
circulo rosado 2,5
circulo rojo 3,5
rectangulo naranja 2,4; 4
```

1) Figuras ordenadas por el tamaño de sus áreas y la cantidad de plástico que se usó por cada una.

Circulo:

```
Circulo color: rojo radio: 2.5 area: 19.6349
Circulo color: rosado radio: 2.5 area: 19.6349
Circulo color: negro radio: 3 area: 28.2743
Circulo color: gris radio: 3.2 area: 32.1699
Circulo color: rojo radio: 3.5 area: 38.4845
La cantidad total de plastico de los circulos es: 138.199
```

Cilindro:

```
Cilindro color: amarillo radio: 2.5 altura: 3 area: 86.3937
Cilindro color: violeta radio: 3 altura: 4 area: 131.947
La cantidad total de plastico de los cilindros es: 218.34
```

Triangulo:

```
Triangulo color: marron cateto mayor: 5 cateto menor: 2.5 area: 6.25
La cantidad total de plastico de los triangulos es: 6.25
```

Cubo:

```
Cubo color: azul lado: 4 area: 96
La cantidad total de plastico de los Cubos es: 96
```

Rectángulo:

```
Rectangulo color: naranja base: 1.4 altura: 2 area: 2.8
Rectangulo color: naranja base: 2.4 altura: 4 area: 9.6
Rectangulo color: verde base: 6 altura: 3 area: 18
La cantidad total de plastico de los rectangulos es: 30.4
```

2) Listar la cantidad de plástico utilizado por figura y color.

Circulo:

```
La cantidad total de plastico de los circulos de color rojo es: 58.1194
La cantidad total de plastico de los circulos de color azul es: 0
La cantidad total de plastico de los circulos de color amarillo es: 0
La cantidad total de plastico de los circulos de color verde es: 0
La cantidad total de plastico de los circulos de color marron es: 0
La cantidad total de plastico de los circulos de color negro es: 28.2743
La cantidad total de plastico de los circulos de color gris es: 32.1699
La cantidad total de plastico de los circulos de color naranja es: 0
La cantidad total de plastico de los circulos de color violeta es: 0
La cantidad total de plastico de los circulos de color rosado es: 19.6349
```

Cilindro:

```
La cantidad total de plastico de los cilindros de color rojo es: 0
La cantidad total de plastico de los cilindros de color azul es: 0
La cantidad total de plastico de los cilindros de color amarillo es: 86.3937
La cantidad total de plastico de los cilindros de color verde es: 0
La cantidad total de plastico de los cilindros de color marron es: 0
La cantidad total de plastico de los cilindros de color negro es: 0
La cantidad total de plastico de los cilindros de color gris es: 0
La cantidad total de plastico de los cilindros de color naranja es: 0
La cantidad total de plastico de los cilindros de color violeta es: 131.947
La cantidad total de plastico de los cilindros de color rosado es: 0
```

Triangulo:

```
La cantidad total de plastico de los triangulos de color rojo es: 0
La cantidad total de plastico de los triangulos de color azul es: 0
La cantidad total de plastico de los triangulos de color amarillo es: 0
La cantidad total de plastico de los triangulos de color verde es: 0
La cantidad total de plastico de los triangulos de color marron es: 6.25
La cantidad total de plastico de los triangulos de color negro es: 0
La cantidad total de plastico de los triangulos de color gris es: 0
La cantidad total de plastico de los triangulos de color naranja es: 0
La cantidad total de plastico de los triangulos de color violeta es: 0
La cantidad total de plastico de los triangulos de color rosado es: 0
```

Cubo:

```
La cantidad total de plastico de los cubos de color rojo es: 0
La cantidad total de plastico de los cubos de color azul es: 96
La cantidad total de plastico de los cubos de color amarillo es: 0
La cantidad total de plastico de los cubos de color verde es: 0
La cantidad total de plastico de los cubos de color marron es: 0
La cantidad total de plastico de los cubos de color negro es: 0
La cantidad total de plastico de los cubos de color gris es: 0
La cantidad total de plastico de los cubos de color naranja es: 0
La cantidad total de plastico de los cubos de color violeta es: 0
La cantidad total de plastico de los cubos de color rosado es: 0
```

Rectángulos:

```
La cantidad total de plastico de los rectangulos de color rojo es: 0
La cantidad total de plastico de los rectangulos de color azul es: 0
La cantidad total de plastico de los rectangulos de color amarillo es: 0
La cantidad total de plastico de los rectangulos de color verde es: 18
La cantidad total de plastico de los rectangulos de color marron es: 0
La cantidad total de plastico de los rectangulos de color negro es: 0
La cantidad total de plastico de los rectangulos de color gris es: 0
La cantidad total de plastico de los rectangulos de color naranja es: 12.4
La cantidad total de plastico de los rectangulos de color violeta es: 0
La cantidad total de plastico de los rectangulos de color rosado es: 0
```

3) Listar dentro de cada color, el orden en que se fueron produciendo las distintas figuras.

C:\Users\Naryadg\Desktop\lectura\bin\Debug\lectura.exe

```
Color rojo por orden de llegada:
1- circulo (2.5)
2- circulo (3.5)

Color azul por orden de llegada:
1- cubo (4)

Color amarillo por orden de llegada:
1- cilindro (2.5; 3)

Color verde por orden de llegada:
1- rectangulo (6; 3)

Color marron por orden de llegada:
1- triangulo (5; 2.5)

Color negro por orden de llegada:
1- circulo (3)

Color gris por orden de llegada:
1- circulo (3.2)

Color naranja por orden de llegada:
1- rectangulo (1.4; 2)
2- rectangulo (2.4; 4)

Color violeta por orden de llegada:
1- cilindro (3; 4)

Color rosado por orden de llegada:
1- circulo (2.5)

Presione una tecla para continuar . . .
```