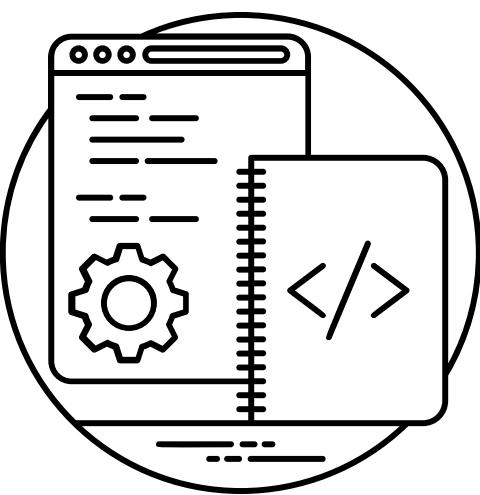




ANTONUCCI JUSTINA MELINSKY JULIETA

PARCIAL 2





PROBLEMA

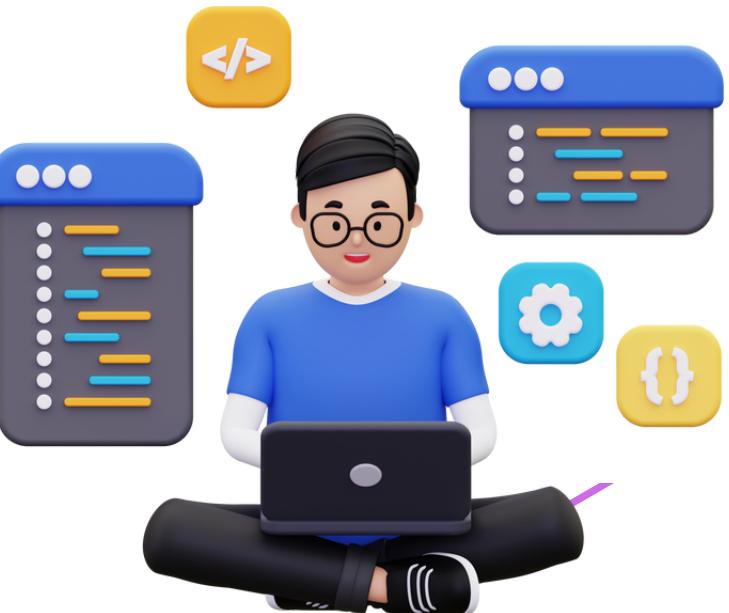


1

Una empresa requiere hacer diariamente el inventariado fisico de una gran variedad de productos, pero estan en un Excel ilegible.

2

Para solucionarlo debiamos realizar un programa con funciones que permitan ver:



La cantidad total de articulos diferentes.

Listado de articulos que estan en el minimo stock.

Listado de articulos que estan en el minimo stock y por deposito.

Los articulos que igualan o superan determinada cantidad de stock.

Cantidad total de articulos.

Stock individual de cada articulo.

Stock individual de cada articulo segun deposito.

LIBRERIAS

<CTIME>

- Proporciona funciones para trabajar con el tiempo y la fecha.
- La utilizamos para medir el tiempo de ejecución de ciertas partes del programa mediante la función clock().

```
#include <ctime>
#include <iostream>
#include <sstream>
#include <fstream>
#include <string>
#include <vector>
#include<stdlib.h>
```

<STDLIB.H>

- Nos sirve para poder crear los arreglos dinamicos (tiene new y delete)

<VECTOR>

- Proporciona una implementación de un contenedor de tipo arreglo dinámico, conocido como std::vector.
- Permite **almacenar** una colección de **elementos de un mismo tipo** y cambiar su tamaño de manera dinámica.

<FSTREAM>

- Proporciona las funciones necesarias para trabajar con archivos de texto y binarios.

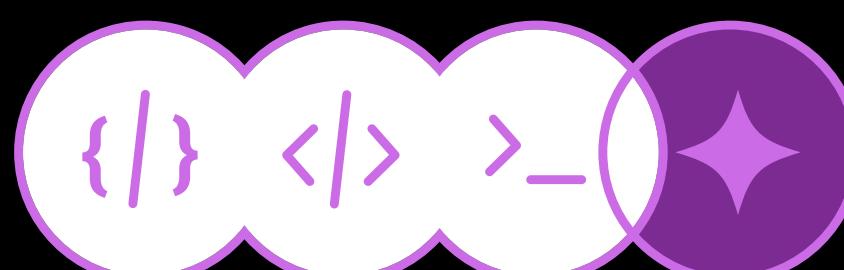
- La utilizamos para abrir y leer un archivo llamado "Inventariado Fisico.csv".

<STRING>

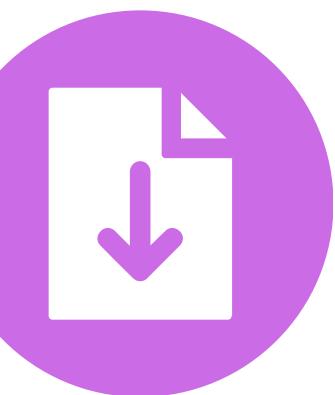
- Proporciona herramientas para manipular y trabajar con cadenas de texto.
- En el código, lo utilizamos para declarar y manipular variables de tipo std::string.

<SSTREAM>

- Permite trabajar con flujos de datos de manera similar a como lo harías con archivos, pero en la memoria.
- Es útil para convertir tipos de datos y manipular cadenas de texto.



ABRIMOS EL ARCHIVO Y NOTIFICA EN CASO DE ERROR



`("Inventariado Fisico.csv", ios::in);`

- Abre el archivo llamado "Inventariado Fisico.csv" en modo lectura (ios::in) y lo asocia con la variable archivo.

`if (archivo.fail())`

- Comprueba si ha habido un fallo al abrir el archivo

`string linea;`

- Declara una variable linea que usamos para almacenar cada línea leída del archivo.

`char delimiter`

- Declara un carácter delimiter que será utilizado para separar los diferentes datos en cada línea del archivo CSV.

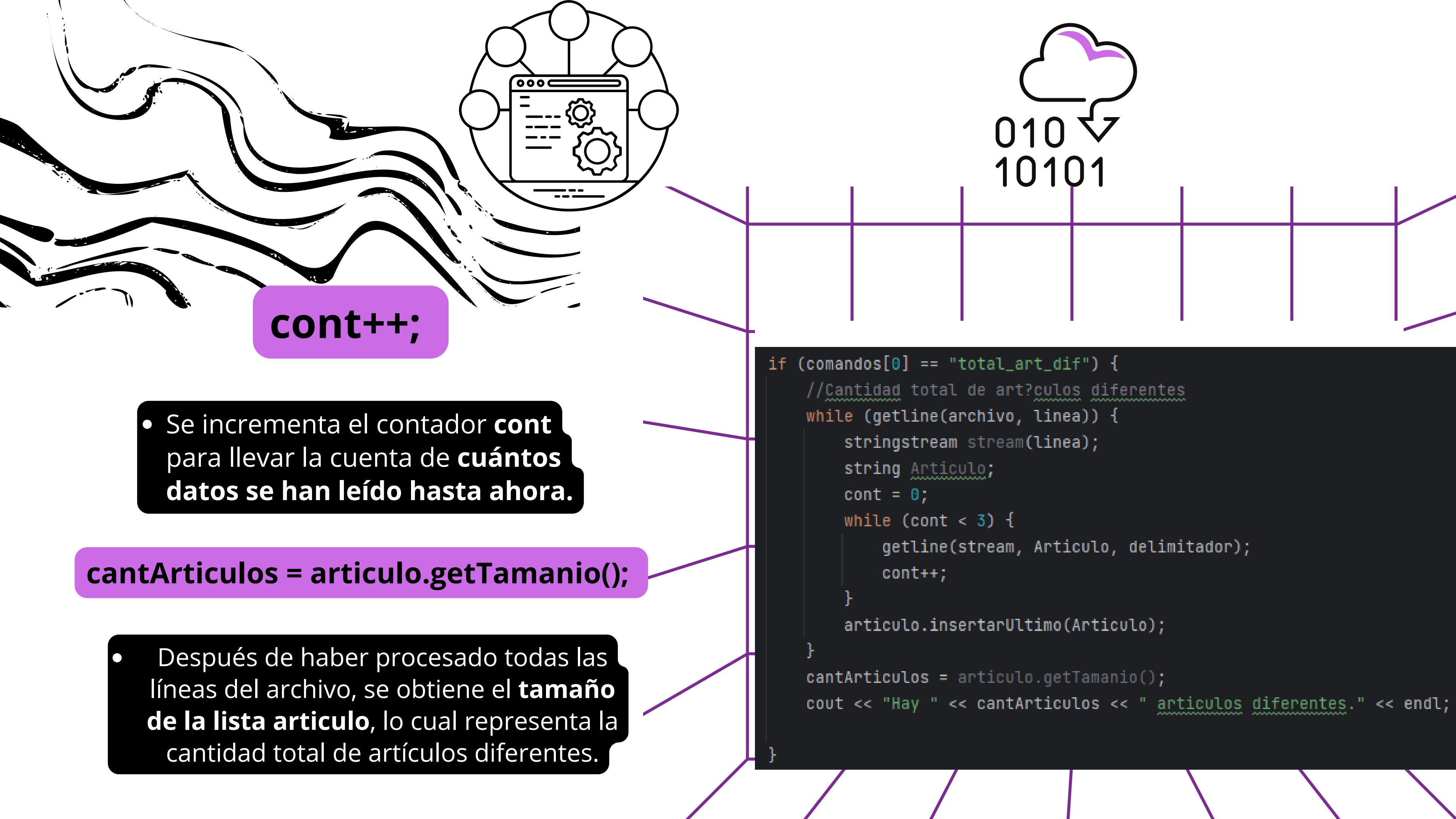
ARCHIVO.OPEN

```
archivo.open("Inventariado Fisico.csv", ios::in);  
if (archivo.fail()) //Notificamos en caso  
    cout << "Error";  
string linea;  
char delimiter = ',';
```

TOTAL ARTÍCULOS DIFERENTES

```
if (comandos[0] == "total_art_dif") {  
    //Cantidad total de artículos diferentes  
    while (getline(archivo, linea)) {  
        stringstream stream(linea);  
        string Articulo;  
        cont = 0;  
        while (cont < 3) {  
            getline(stream, Articulo, delimitador);  
            cont++;  
        }  
        articulo.insertarUltimo(Articulo);  
    }  
    cantArticulos = articulo.getTamanio();  
    cout << "Hay " << cantArticulos << " artículos diferentes." << endl;  
}
```

- 1.Se verifica si el primer elemento de la lista de comandos es "**total_art_dif**".
- 2.Si se cumple la condición anterior, se procede a realizar la función:
 - Se inicia un **bucle while** para leer cada línea del archivo.
 - Para cada línea, se utiliza un **stringstream** para procesarla.
 - Se inicializa una variable '**Articulo**' para almacenar el nombre del artículo.
 - Se utiliza un bucle while con la condición '**cont < 3**' para extraer el tercer elemento de la línea (que representa el nombre del artículo).
 - El nombre del artículo se inserta en la **estructura 'articulo'**.
 - Al finalizar el procesamiento, se obtiene la **cantidad de elementos** en la estructura 'articulo' y se almacena en la variable '**cantArticulos**'.
 - Se **imprime** el mensaje indicando la cantidad total de artículos diferentes.



```
if (comandos[0] == "total_art_dif") {
    //Cantidad total de artículos diferentes
    while (getline(archivo, linea)) {
        stringstream stream(linea);
        string Artículo;
        cont = 0;
        while (cont < 3) {
            getline(stream, Artículo, delimitador);
            cont++;
        }
        articulo.insertarUltimo(Artículo);
    }
    cantArticulos = articulo.getTamano();
    cout << "Hay " << cantArticulos << " artículos diferentes." << endl;
}

if (comandos[0] == "total_art") {
    //Cantidad total de artículos
    while (getline(archivo, linea)) {
        stringstream stream(linea);
        string depositos;
        cont = 0;
        while (getline(stream, depositos, delimitador)) {
            if (cont >= 3)
                Depositos.insertarUltimo(depositos);
            cont++;
        }
        sumaProductos += Depositos.sumarDeposito();
        Depositos.vaciar();
    }
    cout << "Hay " << sumaProductos << " artículos en total." << endl;
}
```

TOTAL ARTICULOS

Función: Calcular la cantidad total de artículos en un archivo

- La función se activa cuando el primer elemento de la lista de comandos es "**total_art**".
- Se utiliza un **bucle while** para leer cada línea del archivo.
- Se crea un **stringstream** para procesar la línea actual.
- Se inicializa una variable '**cont**' para llevar un conteo de los elementos procesados.
- Otro bucle while se utiliza para separar los elementos de la línea utilizando un delimitador.

1. Si el contador 'cont' es mayor o igual a 3, se inserta el elemento en la estructura 'Depositos'.

2. Se suma el valor del depósito actual a la variable 'sumaProductos'.

3. Se vacía la estructura 'Depositos' para prepararla para la siguiente línea.

4. Al finalizar el procesamiento, se imprime el total de artículos encontrado.

```
if (comandos[0] == "min_stock" && comandos.size() == 2) {  
    //Listado de artículos con cantidad n o menos de stock.  
    min = stoi(comandos[1]);  
    arregloPosiciones = new int[cantidadObjetos];  
  
    while (getline(archivo, linea)) {  
        stringstream stream(linea);  
        string Articulo, depositos;  
        cont = 0;  
        while (getline(stream, depositos, delimitador)) {  
            if (cont == 2)  
                articulo.insertarUltimo(depositos);  
            if (cont >= 3)  
                Depositos.insertarUltimo(depositos);  
            cont++;  
        }  
        sumaProductos = Depositos.sumarDeposito();  
        if (sumaProductos <= min) {  
            arregloPosiciones[i] = contador;  
            i++;  
        }  
        contador++;  
        Depositos.vaciar();  
    }  
    i = 0;  
    while (arregloPosiciones[i] >= 0) {  
        longitud++;  
        i++;  
    }  
    posicion = new int[longitud];  
    for (i = 0; i < longitud; i++)  
        posicion[i] = arregloPosiciones[i];  
  
    articulo.getData( pos: posicion, longi: longitud, cantidad: min);  
}
```

1 Se verifica si el primer elemento de la lista de comandos es "min_stock" y si el tamaño de la lista de comandos es 2 o 3.

2 Si se cumplen las condiciones anteriores, se procede a realizar la función correspondiente:

a) En el primer caso (comandos.size() == 2), se espera un valor de stock mínimo. Este valor se convierte a entero y se almacena en la variable 'min'.

b) Se inicializa un arreglo de posiciones y se lee el archivo línea por línea.

c) Se utiliza un stringstream para procesar la línea actual y se extraen los elementos necesarios (nombre del artículo y cantidades de depósitos).

d) Se comprueba si la cantidad de stock es menor o igual a 'min'. Si es así, se guarda la posición del artículo en el arreglo de posiciones.

MINIMO STOCK

```

if (comandos[0] == "min_stock" && comandos.size() == 3) {
    //Listado de artículos con cantidad n o menos de stock según un depósito
    min = std::stoi(comandos[1]);
    eleccionDeposito = std::stoi(comandos[2]);

    if (min < 0)
        cout << "El stock no puede ser menor a 0" << endl;
    else if (eleccionDeposito < 0 && eleccionDeposito <= contadorAncho - 2)
        cout << "No existe el depósito que esta buscando" << endl;
    else {
        arregloPosiciones = new int[cantidadObjetos];

        while (getline(archivo, linea)) {
            stringstream stream(linea);
            string Articulo, depositos;
            cont = 0;
            while (getline(stream, depositos, delimitador)) {
                if (cont == 2)
                    articulo.insertarUltimo(depositos);
                if (cont == 3 + eleccionDeposito - 1)
                    Depositos.insertarUltimo(depositos);
                cont++;
            }

            sumaProductos = Depositos.sumarDeposito();
            if (sumaProductos <= min) {
                arregloPosiciones[i] = contador;
                i++;
            }
            contador++;
            Depositos.vaciar();
        }
        i = 0;
        while (arregloPosiciones[i] >= 0) {
            longitud++;
            i++;
        }
        posicion = new int[longitud];
        for (i = 0; i < longitud; i++)
            posicion[i] = arregloPosiciones[i];
    }
}

articulo.getDataPorDeposito( pos: posicion, longitud, cantidad: min, dep: eleccionDeposito);

```

e)

Al finalizar el procesamiento, se crea un nuevo arreglo de posiciones con la longitud adecuada y se copian los valores relevantes.

f)

Se llama a la función 'getData' para obtener la información de los artículos.

3)

En el segundo caso (comandos.size() == 3), se espera además un valor de elección de depósito. Ambos valores se convierten a enteros y se almacenan en 'min' y 'eleccionDeposito' respectivamente.

4)

Se realizan verificaciones adicionales para asegurarse de que los valores sean válidos.

5)

Se procede de manera similar al primer caso, con la diferencia de que se verifica el depósito específico seleccionado antes de insertar los datos en la estructura 'Depositos'.

6)

Al finalizar el procesamiento, se realiza la misma operación de creación y copia de arreglos de posiciones, y se llama a la función 'getDataPorDeposito' para obtener la información de los artículos según el depósito seleccionado.



STOCK

```
else if (comandos[0] == "stock" && comandos.size() == 3) {  
    //Stock del articulo en un deposito  
    artic = comandos[1];  
    eleccionDeposito = std::stoi(comandos[2]);  
    bool comp = false;  
  
    while (getline(archivo, linea)) {  
        stringstream stream(linea);  
        string depositos, resp;  
        cont = 0;  
        while (getline(stream, depositos, delimitador) || cont <= 7) {  
            if (cont == 2)  
                if (depositos == artic)  
                    resp = depositos;  
            if (cont == (3 + eleccionDeposito - 1) && resp == artic) {  
                Depositos.insertarUltimo(depositos);  
                comp = true;  
            }  
            cont++;  
        }  
        if (comp) {  
            sumaProductos = Depositos.sumarDeposito();  
            break;  
        }  
        contador++;  
    }  
    if (comp == false)  
        cout << "El articulo que busca no existe" << endl;  
    else  
        cout << "El stock de '" << artic << "' en el deposito "  
            << eleccionDeposito << " es: " << sumaProductos << endl;
```

1)

Se verifica si el primer elemento de la lista de comandos es "stock" y si el tamaño de la lista de comandos es 2 o 3.

2)

Si se cumple la condición anterior, se procede a realizar la función correspondiente:

a)

En el primer caso (comandos.size() == 2), se espera un nombre de artículo como argumento. Este valor se almacena en la variable 'artic'.

b)

Se inicializa una variable 'comp' como falsa.

c)

Se lee el archivo línea por línea y se procesa cada línea con un stringstream.

d)

Se extraen los elementos necesarios (nombre del artículo y cantidades de depósitos) y se verifica si coinciden con el artículo buscado.

e)

Si se encuentra el artículo, se insertan las cantidades de depósitos en la estructura 'Depositos' y se marca 'comp' como verdadero.

f)

Se calcula el stock total utilizando 'sumarDeposito' y se interrumpe el bucle.

g)

Si 'comp' sigue siendo falso al finalizar el procesamiento, se imprime un mensaje indicando que el artículo no fue encontrado.

h)

En caso contrario, se muestra el stock total del artículo.

STOCK



```
else if (comandos[0] == "stock" && comandos.size() == 3) {  
    //Stock del articulo en un deposito  
    artic = comandos[1];  
    eleccionDeposito = std::stoi(comandos[2]);  
    bool comp = false;  
  
    while (getline(archivo, linea)) {  
        stringstream stream(linea);  
        string depositos, resp;  
        cont = 0;  
        while (getline(stream, depositos, delimitador) || cont <= 7) {  
            if (cont == 2)  
                if (depositos == artic)  
                    resp = depositos;  
            if (cont == (3 + eleccionDeposito - 1) && resp == artic) {  
                Depositos.insertarUltimo(depositos);  
                comp = true;  
            }  
            cont++;  
        }  
        if (comp) {  
            sumaProductos = Depositos.sumarDeposito();  
            break;  
        }  
        contador++;  
    }  
    if (comp == false)  
        cout << "El articulo que busca no existe" << endl;  
    else  
        cout << "El stock de '" << artic << "' en el deposito "  
            << eleccionDeposito << " es: " << sumaProductos << endl;
```

3)

En el segundo caso (comandos.size() == 3), además del nombre de artículo, se espera un número de depósito como argumento. Ambos valores se almacenan en 'artic' y 'elecciónDepósito' respectivamente.

4)

Se procede de manera similar al primer caso, pero esta vez se verifica si el número de depósito coincide con el seleccionado antes de insertar los datos en la estructura 'Depositos'.

5)

Al finalizar el procesamiento, se calcula el stock del artículo en el depósito especificado y se imprime el resultado.

6)

Si el artículo no se encuentra en el depósito seleccionado, se imprime un mensaje indicando que el artículo no fue encontrado.



MAX STOCK

- 1) Se verifica si el primer elemento de la lista de comandos es "max_Stock".
- 2) Si se cumple la condición anterior, se procede a realizar la función:
 - a) Se imprime el valor de comandos[1] (para verificar el número n que se utilizará como criterio).
 - b) El valor de comandos[1] se convierte a entero y se almacena en la variable 'min'.
 - c) Se inicializa un arreglo de posiciones.
 - d) Se lee el archivo línea por línea y se procesa cada línea con un stringstream.

- e) Se extraen los elementos necesarios (nombre del artículo y cantidades de depósitos) y se insertan en las estructuras 'articulo' y 'Depositos'.
- f) Se calcula el stock total utilizando 'sumarDeposito'.
- g) Si el stock total es igual o supera el valor mínimo 'min', se guarda la posición del artículo en el arreglo de posiciones.
- h) Al finalizar el procesamiento, se crea un nuevo arreglo de posiciones con la longitud adecuada y se copian los valores relevantes.
- i) Se llama a la función 'getdato' para obtener la información de los artículos que cumplen con el criterio de stock.
- j) Se retorna 0 (indicando que la función terminó correctamente).

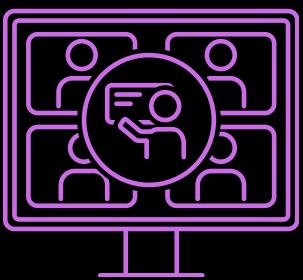
```
if (comandos[0] == "max_Stock") {  
    //Listado de articulos cuyo stock es igual o supera el numero n  
    cout << "comandos[1]: " << comandos[1] << endl;  
  
    min = std::stoi(comandos[1]);  
  
    arregloPosiciones = new int[cantidadObjetos];  
  
    while (getline(archivo, linea)) {  
        stringstream stream(linea);  
        string Articulo, depositos;  
        cont = 0;  
        while (getline(stream, depositos, delimitador)) {  
            if (cont == 2)  
                articulo.insertarUltimo(depositos);  
            if (cont >= 3)  
                Depositos.insertarUltimo(depositos);  
            cont++;  
        }  
        sumaProductos = Depositos.sumarDeposito();  
        if (sumaProductos >= min) {  
            arregloPosiciones[i] = contador;  
            i++;  
        }  
        contador++;  
        Depositos.vaciar();  
    }  
}
```

¿Cómo nos comunicamos? ¿Cómo nos distribuimos el trabajo?



Una de las más grandes complicaciones del proyecto fue la comunicación, se hizo muy difícil coordinar horarios entre cuatro personas en una época llena de parciales y además diferencia horaria que puede parecer poca pero hace una gran diferencia.

Esto puede generar dificultades para encontrar momentos convenientes para reunirse o para resolver problemas en conjunto.



No utilizamos ningún método de comunicación llamativo, solo discord y meet.

Nos distribuimos el trabajo más o menos equitativo, ya que algunas funciones eran un poco más complicadas que otras, hubo algunos problemas de comunicación en cuanto a la entrega de cada uno, y eso desató otros problemas relacionados al tiempo de trabajo y distribución del mismo.



¿Qué opinamos de la experiencia?

Estamos de acuerdo en que esta experiencia fue **valiosa y enriquecedora**, especialmente dada la naturaleza de nuestra carrera. Entendemos que enfrentar **desafíos de comunicación** y coordinación es algo que probablemente se repita en el futuro, y verlo como una **oportunidad** de aprendizaje es fundamental.

Aprender de los errores cometidos en este proyecto nos brinda una base sólida para evitarlos en **futuros trabajos**, lo cual es muy beneficioso para nuestro crecimiento profesional.



El tema de la dificultad para coordinar horarios y el **uso de plataformas diferentes** fue un desafío real. Sin embargo, también reconocemos que esto formó parte de nuestro aprendizaje. Aprender a adaptarnos y trabajar eficientemente en entornos diversos y con herramientas variadas es una habilidad valiosa en el mundo laboral actual. A pesar de que pueda haber **ralentizado el proceso**, creemos que esta experiencia nos ha preparado mejor para enfrentar situaciones similares en el futuro.



Muchas Gracias!

*