



UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería de Sistemas

**SOLUCIÓN DE GESTIÓN DE INVENTARIOS
AUTOMATIZADA PARA PEQUEÑAS EMPRESAS**

Curso: Estructura de Datos

Docente: MSc. Haydee Raquel Sisa Yataco

INTEGRANTES:

Calloticona Chambilla, Marymar Danytza	(2023076791)
Ramos Loza, Mariela Estefany	(2023077478)
Carbajal Vargas, Andre Alejandro	(2023077287)
Yupa Gómez Fátima Sofía	(2023076618)

Tacna – Perú

2024

Índice General

INTRODUCCIÓN.....	4
SOLUCIÓN DE GESTIÓN DE INVENTARIOS AUTOMATIZADA PARA PEQUEÑAS EMPRESAS.....	5
I. Planteamiento del problema.....	5
A. Justificación.....	6
B. Alcance.....	6
II. Objetivo.....	7
A. General:.....	7
B. Específicos:.....	8
III. Marco Teórico.....	8
A. Estructura de control condicional dobles:.....	8
B. Estructura de control condicional múltiples:.....	9
C. Pair:.....	9
D. Ficheros:.....	10
E. Estructuras repetitivas:.....	11
1.1 Do while.....	11
1.2 While.....	11
1.3 For.....	12
F. Registros:.....	12
G. Búsqueda Binaria:.....	13
H. Método burbuja:.....	13
I. Funciones:.....	14
IV. Desarrollo de la Solución.....	15
A. Lenguaje de implementación:.....	15
B. Estructuras a emplear:.....	15
C. Características del sistema a implementar:.....	15
VI. CONCLUSIONES.....	19

VII. RECOMENDACIONES PARA EL USO DEL PROGRAMA.....	20
Bibliografía:.....	21
VIII. Anexos.....	22
A. Imágenes del proyecto.....	22
IX. DÍPTICO:.....	23
A. Video demo del proyecto.....	24

INTRODUCCIÓN

En el contexto de una pequeña empresa, donde cada recurso y decisión cuenta para el éxito del negocio, la gestión efectiva del inventario juega un papel crucial. Un programa de control de inventario desarrollado en C++ se convierte en una herramienta invaluable para asegurar que los productos estén disponibles cuando los clientes los necesitan, minimizando al mismo tiempo los costos de almacenamiento y maximizando el potencial de ingresos.

Este proyecto se centra en crear una solución personalizada y accesible que satisfaga las necesidades específicas de una pequeña empresa. Desde la capacidad de mantener un registro detallado y actualizado de todos los productos disponibles hasta la generación de informes analíticos que ayuden en la toma de decisiones estratégicas, el programa en C++ será diseñado para ser intuitivo, eficiente y fácil de usar.

Utilizando principios sólidos de programación, como estructuras de datos eficientes para la gestión de inventarios, algoritmos optimizados para la búsqueda y la clasificación, y técnicas de gestión de memoria que optimicen los recursos disponibles, nuestro objetivo es ofrecer una solución tecnológica que no solo mejore la organización interna, sino que también impulse la productividad y la competitividad de la empresa en el mercado.

Además de su función práctica, este proyecto representa una oportunidad emocionante para aplicar nuestra experiencia en desarrollo de software de manera directa y significativa. Estamos comprometidos a proporcionar a los propietarios de pequeñas empresas una herramienta poderosa que no solo simplifique la gestión del inventario, sino que también contribuya al crecimiento sostenible y al éxito continuo del negocio en un entorno competitivo.

SOLUCIÓN DE GESTIÓN DE INVENTARIOS AUTOMATIZADA PARA PEQUEÑAS EMPRESAS

I. Planteamiento del problema

En el entorno de las pequeñas y medianas empresas, la gestión del inventario es una tarea crítica para asegurar la continuidad del negocio y la satisfacción del cliente. Sin embargo, muchas de estas empresas aún dependen de métodos manuales para llevar el control de sus productos, lo que a menudo resulta en problemas significativos que afectan su operación diaria.

Estos métodos son propensos a errores y consumen mucho tiempo, lo que impide obtener datos en tiempo real. Como resultado, las empresas enfrentan faltantes de productos, excesos de inventario y dificultades en la toma de decisiones. Los errores en el registro pueden provocar faltantes de productos, afectando la satisfacción del cliente y llevando a la pérdida de ventas. Además, la gestión ineficaz del inventario puede resultar en excesos de stock, generando costos adicionales de almacenamiento y obsolescencia de productos.

Es crucial desarrollar un sistema automatizado que reduzca los errores humanos y ahorre tiempo, proporcionando datos en tiempo real. Un programa bien diseñado en C++ permitirá una gestión más eficiente del inventario, mejorará la precisión de las entradas y salidas de productos, y facilitará la generación de informes útiles para la toma de decisiones.

Las pequeñas y medianas empresas enfrentan desafíos significativos en la gestión manual del inventario, que resulta en errores, ineficiencias y pérdidas económicas.

A. Justificación

Implementar un programa de gestión de inventario en C++ permitirá a las empresas manejar sus productos de manera más precisa y eficiente, reduciendo los errores humanos y optimizando el uso de recursos.

B. Alcance

El programa cubrirá las funcionalidades de registro de productos, control de entradas y salidas, generación de informes y una interfaz de usuario intuitiva y segura.

El proyecto permite la gestión de usuarios mediante un sistema de inicio de sesión y registro de nuevos usuarios, asignando automáticamente el rol predeterminado de usuario. Para mejorar la seguridad, las contraseñas no serán visibles durante su ingreso. El programa permite la modificación de tres tipos de inventarios: Bebidas, Snacks y Abarrotes, ofreciendo funcionalidades para mostrar productos, insertar nuevos, buscar por ID, actualizar, eliminar y, finalmente, guardar y salir. Todas las modificaciones se registran en un archivo Excel, proporcionando una visión clara y detallada del inventario. Cada sección del inventario gestionará atributos como el ID del producto, nombre, costo, cantidad y, en el caso de las bebidas, su volumen en mililitros.

Sin embargo, es importante considerar ciertas limitaciones del programa para su implementación efectiva en entornos específicos. El programa no registra el IGV (Impuesto General a las Ventas), lo cual es crucial para cumplir con las obligaciones fiscales y contables requeridas por la normativa vigente en muchas jurisdicciones. Además, presenta limitaciones en la gestión de usuarios, permitiendo únicamente la creación de usuarios sin proporcionar funcionalidades

para la eliminación o actualización de credenciales. Esto puede limitar la seguridad y la administración eficiente de acceso al programa.

Otra limitación significativa es que el programa puede no ser funcional para empresas con gran volumen de inventario, debido a sus capacidades limitadas de almacenamiento y procesamiento de datos. La falta de optimización en la gestión de grandes cantidades de productos podría afectar negativamente la eficiencia operativa y la capacidad de respuesta ante la demanda del mercado. Adicionalmente, el programa no proporciona una función para calcular y mostrar un total de inventario, lo cual puede dificultar la gestión y el control de los productos almacenados.

Asimismo, el programa no está diseñado para avisar si hay escasez de algún producto, lo que puede llevar a situaciones de falta de stock y afectar la capacidad de respuesta ante la demanda del mercado. También carece de la capacidad de registrar la fecha de caducidad de los productos, lo que puede llevar a la venta de productos vencidos y a la pérdida de productos que han caducado.

Estas limitaciones deben ser consideradas cuidadosamente para asegurar que el programa se implemente de manera efectiva y cumpla con las necesidades específicas del entorno en el que será utilizado.

II. Objetivo

A. General:

Desarrollar un programa de gestión de inventario utilizando el lenguaje de programación C++, el cual nos permite realizar el mantenimiento del inventario de una entidad.

B. Específicos:

Diseñar e implementar un programa de inicio de sesión seguro y confiable para garantizar el acceso controlado al programa.

- Desarrollar un menú de opciones intuitivo y fácil de usar que permita al usuario navegar por las diferentes funcionalidades del programa.
- Crear una interfaz gráfica amigable y visualmente atractiva que facilite la interacción del usuario con el programa.
- Implementar las funcionalidades de inserción, actualización, eliminación y consulta de registros de la entidad seleccionada, utilizando las estructuras de datos y algoritmos apropiados.
- Documentar detalladamente el proceso de desarrollo, incluyendo diagramas de flujo y código fuente.

III. Marco Teórico

Este informe abarca diversos conceptos fundamentales de la programación, destacando el conocimiento de la sintaxis y las estructuras de control del lenguaje C++. En este contexto, se hace énfasis en el uso de arreglos, registros y listas, estos elementos son imprescindibles para organizar y manipular la información de manera básica.

A. Estructura de control condicional dobles:

La estructura SÍ-SINO tiene una composición similar a la del St; la diferencia está en que el SI-SINO tiene instrucciones tanto para el valor verdadero de la condición como para el falso.




```
if(Condición)
    Acción por ejecutar
else
    Acción por ejecutar
```


Si la condición es verdadera, ejecuta las instrucciones de la línea 2; pero si es falsa, el flujo de datos omite estas instrucciones y ejecuta de manera automática las de la línea 4, es decir, entra de una vez al SINO.

B. Estructura de control condicional múltiples:

La sentencia switch permite realizar una selección múltiple. En primer lugar, se evalúa la expresión, que debe aparecer entre paréntesis. La expresión debe producir un resultado de tipo entero, carácter o lógico. A continuación se compara el resultado de evaluar la expresión con los literales que aparecen en los casos. Si el resultado de evaluar la expresión coincide con el literal de algún caso, entonces se ejecuta el conjunto de instrucciones asociado al caso y el conjunto de instrucciones asociadas al resto de casos que aparecen con posterioridad hasta que se ejecuta la instrucción break-que provoca que se salga de la ejecución de la instrucción switch-o se han ejecutado todos los casos. Por lo tanto, si para un caso sólo queremos que se ejecute un conjunto de instrucciones debemos terminar ese conjunto de instrucciones con la sentencia break-la sentencia break es opcional. La etiqueta default representa un caso que se ejecuta si el resultado de evaluar la expresión no coincide con ninguno de los literales de los casos. El especificar una etiqueta default es opcional.



```
switch(variable de control){  
    case 1: codigo a ejecutar;  
        break;  
    case 2: codigo a ejecutar;  
        break;  
    default: codigo a ejecutar;  
        break;  
}
```

C. Pair:

Una estructura que proporciona la capacidad de tratar dos objetos como uno solo.

Parámetros:

Val1

Valor que inicializa el primer elemento de pair.

Val2

Valor que inicializa el segundo elemento de pair.

Derecho

Un par cuyos valores se deben usar para inicializar los elementos de otro par.

El primer constructor (predeterminado) inicializa el primer elemento del par en el valor predeterminado del tipo T1 y el segundo elemento en el valor predeterminado del tipo T2. Se define si ambos tipos son `default{>-<}` constructible de forma predeterminada.

El segundo constructor inicializa el primer elemento del par en Val1 y el segundo, en Val2. Se define si ambos tipos son copy constructible.

El tercer constructor (plantilla) inicializa el primer elemento del par en `Right. first` y el segundo en `Right. second`. Se define si ambos tipos del par se pueden construir a partir de los tipos de valor proporcionados.

El cuarto constructor inicializa el primer elemento del par en Val1 y el segundo en Val2 mediante el Declarador de referencia de Rvalue: `&&`. Se define si ambos tipos del par se pueden construir a partir de los tipos de valor proporcionados.

D. Ficheros:

En la mayor parte de los programas se quiere almacenar información que no desaparezca al finalizar la ejecución de los mismos. Esto es lo que diferencia el almacenamiento de datos en ficheros al almacenamiento de datos en estructuras, vectores, etc... Antiguamente el C trataba bajo Unix los ficheros como una sucesión de octetos (grupos de 8 bits) pudiendo accederse a cada uno de forma individual. En posteriores generalizaciones del C en sistemas operativos diferentes se modificó la forma de tratar y usar los ficheros por parte del C, creándose una serie de librerías para facilitar su manipulación. Este es el caso de la librería `stdio.h` (acrónimo de standard input/output), que

define estructuras de tipo fichero y un conjunto de funciones que nos facilitarán el trabajo con los ficheros.

E. Estructuras repetitivas:

1.1 Do while

En esta estructura do-while, la condición de continuación del ciclo se prueba al final del mismo. Funciona de manera similar a la estructura while; la diferencia es que una evalúa al inicio del ciclo y la otra al final. En esta estructura es indispensable escribir las llaves, aunque pudiera parecer innecesario utilizarlas. También se debe notar que al final de la condición do-while se escribe punto y coma. La representación de la estructura es la siguiente:



```
do {  
    // Código que se ejecuta al menos una vez  
} while (condición);
```

1.2 While

En esta estructura, la repetición se realizará tantas veces como se indique mientras se cumpla una condición. La cantidad de repeticiones puede ser definida o indefinida. La representación de la estructura es la siguiente:



```
while (condición) {  
    // Código que se ejecuta mientras la condición sea verdadera  
}
```

1.3 For

La estructura de control “for” se utiliza generalmente cuando la repetición está definida. Esta estructura maneja todos los detalles de la repetición controlada por el contador. La representación de la estructura repetitiva es la siguiente:

```
● ● ●  
for (inicialización; condición; actualización) {  
    // Código que se ejecuta mientras la condición sea verdadera  
}
```

F. Registros:

Una de las estructuras más utilizadas en lenguaje de programación es struct. Estas permiten encapsular campos o atributos dentro de un nombre de estructura y luego poder usarlas, proporcionando una manera más organizada y clara de manejar información compleja. En la implementación de un programa de gestión de almacén, los registros son fundamentales para representar entidades como productos, clientes o transacciones. Por ejemplo, una estructura de cuenta podría estar conformada por su número, su fecha de inscripción, su monto original de apertura, su saldo, entre otros atributos. Luego, esa estructura se utilizaba como un todo, pero se podía acceder a cada atributo particular.

```
● ● ●  
// Definición de una estructura  
struct NombreEstructura {  
    tipo1 miembro1;  
    tipo2 miembro2;  
    // ...  
    tipoN miembroN;  
};
```

G. Búsqueda Binaria:

Una búsqueda binaria típica es la búsqueda de una palabra en un diccionario. Dada la palabra, se abre el libro cerca del principio, del centro o del final dependiendo de la primera letra del primer apellido o de la palabra que busca. Se puede tener suerte y acertar con la página correcta, pero normalmente no será así y se mueve el lector a la página anterior o posterior del libro. Por ejemplo, si la palabra comienza con "J" y se está en la "L" se mueve uno hacia atrás. El proceso continúa hasta que se encuentra la página buscada o hasta que se descubre que la palabra no está en la lista.

```
int busquedaBin(int lista[], int n, int clave){
    int central, bajo, alto;
    int valorCentral;
    bajo=0;
    alto= n-1;
    while (bajo <= alto) {
        central (bajo + alto)/2;
        valorCentral lista[central];
        if (clave == valorCentral) return central;
        else if (clave < valorCentral) alto central -1;
        else
            bajo central + 1;
    }
    return -1;
}
```

H. Método burbuja:

La técnica utilizada se denomina ordenación por burbuja u ordenación por hundimiento debido a que los valores más pequeños “burbujean” gradualmente (suben) hacia la cima o parte superior del array de modo similar a como suben las burbujas en el agua, mientras que los valores mayores se hunden en la parte inferior del array. La técnica consiste en hacer varias pasadas a través del array. En cada pasada, se comparan parejas sucesivas de elementos. Si una pareja está en orden creciente (o los valores son idénticos), se dejan los valores como están. Si una pareja está en orden decreciente, sus valores se intercambian en el array.



```
void ordenamientoBurbuja(Tipo datos[], int n) {  
    for (int i = 0; i < n-1; i++) {  
        for (int j = 0; j < n-i-1; j++) {  
            if (datos[j] > datos[j+1]) {  
                // Intercambiar datos[j] y datos[j+1]  
                Tipo temp = datos[j];  
                datos[j] = datos[j+1];  
                datos[j+1] = temp;  
            }  
        }  
    }  
}
```

I. Funciones:

La declaración de una función nos da el nombre de la función, el tipo del valor que retorna y el número y tipo de parámetros que deben pasárselo. La sintaxis es:



```
tipo_retorno nom_funcion(lista_tipos_param){  
    lista_tipos_param = tipo_param_1; tipo_param_2, ... ; tipo_param_n;  
}
```

Donde los paréntesis y el punto y coma son obligatorios. El tipo de retorno se puede omitir pero el compilador asume que es int. En C una función sin lista de parámetros se considera que tiene un número de parámetros indefinidos, mientras que en C++ se entiende que no se le pasa nada (para pasar un número de parámetros indefinido se ponen tres puntos (...) en la lista de parámetros). Lo más recomendable para evitar confusiones es poner siempre void como parámetro cuando no vamos a pasar nada.

En conclusión, la implementación de un programa en C++ para la gestión de un almacén de negocios se apoya en la sintaxis y las estructuras de control del lenguaje, el uso de arreglos y registros, las estructuras de datos, y las técnicas de optimización y gestión de memoria. Este enfoque integral permite a los desarrolladores crear soluciones informáticas eficientes y de alta calidad para la gestión de almacenes.

IV. Desarrollo de la Solución

A. Lenguaje de implementación:

Este proyecto está siendo realizado con código de C++, un lenguaje de programación suficiente para realizar un programade gestión de inventario básico para un negocio, pues cuenta con una extensa biblioteca estándar que proporciona una amplia gama de funcionalidades, desde estructuras de datos hasta algoritmos de procesamiento de datos. Esto nos permite aprovechar estas herramientas para acelerar el desarrollo del programade gestión de inventario, sin tener que implementar todo desde cero.

B. Estructuras a emplear:

- Estructuras de control de datos secuenciales.
- Estructuras de control condicionales simples.
- Estructuras de control condicionales dobles.
- Estructuras de control condicionales múltiples.
- Estructuras de control repetitivas (Do while, while, for).
- Registros.
- Algoritmo de ordenación (método burbuja).
- Algoritmo de búsqueda binaria.

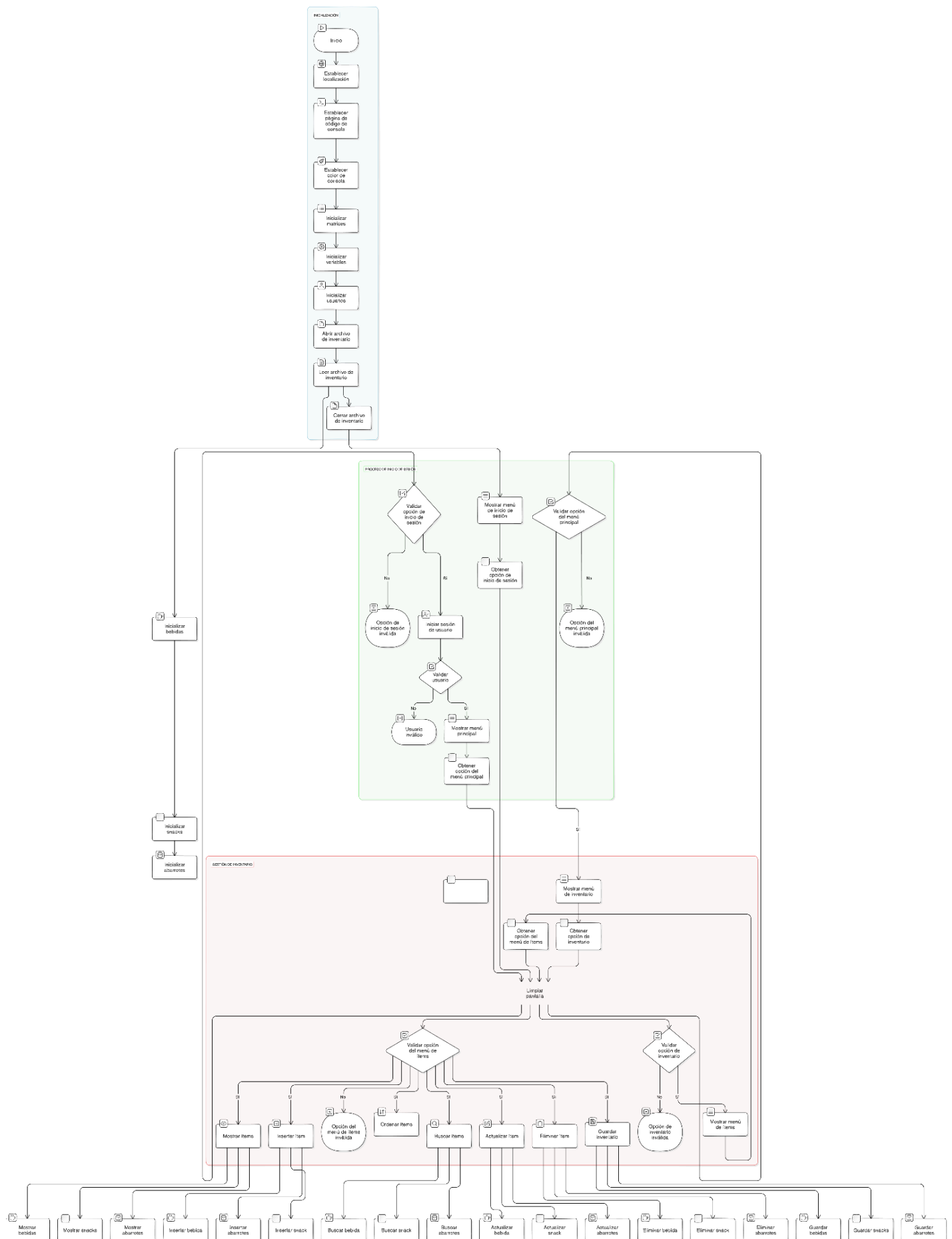
C. Características del sistema a implementar:

- Para la ejecución del programa utilizamos la versión 6.3 del DEV C++.
- Además utilizamos las siguientes librerías:
 - <iostream>: Maneja la entrada y salida de datos.
 - <locale.h>: Configura la localización para manejar adecuadamente formatos y caracteres específicos de distintos idiomas.

- `<wchar.h>`: Proporciona funciones para trabajar con caracteres anchos, permitiendo el manejo de una gama más amplia de caracteres.
- `<windows.h>` y `<stdio.h>`: Nos permite mover el cursor a una ubicación específica en la pantalla y suspender la ejecución del programa por un tiempo específico.
- `<limits>`: Define propiedades y límites de tipos de datos aritméticos, proporcionando información sobre los valores mínimo y máximo que pueden tener estos tipos.
- `<strings>`: Proporciona funciones para trabajar con cadenas de caracteres (strings).
- `<cstring>` : En C++, las cadenas de caracteres (c-strings) y las funciones de manejo de cadenas de la biblioteca estándar de C, como `strcpy` y `strtok`, se pueden utilizar mediante la inclusión de la biblioteca `<cstring>`. Aquí tienes un ejemplo que muestra cómo utilizar estas funciones en C++
- `<stdio.h>`: Nos permite realizar operaciones con archivos.
- Dentro de la función `main()`, se configura la localización a `"es_ES.UTF-8"` para manejar caracteres en español. Además, se cambia el color de la consola a blanco con texto azul mediante el comando `system("COLOR F1")`.
- Dentro del archivo `persona.h` se declara un registro llamado `persona`, para que guarde los datos del usuario (nombre de usuario, contraseña y rol).
- El programa entra en un bucle principal donde se muestra el menú de inicio de sesión. Dependiendo de la opción seleccionada, el usuario puede intentar iniciar sesión, registrarse o salir del programa. Si el usuario inicia sesión correctamente, se muestra un menú de gestión de inventario. Donde podrá gestionar bebidas, snacks o abarrotes, para ello se declaran tres registros más, para almacenar los diferentes tipos de productos.
- Ahora debemos saber que existen dos tipos de roles y que respecto a eso le permitirá al usuario realizar distintas funciones.

- Administradores: Pueden realizar todas las operaciones sobre los productos (mostrar, insertar, buscar, actualizar y eliminar).
- Empleados: Pueden realizar un conjunto limitado de operaciones, como ver y buscar productos.
- Ahora debemos iniciar una estructura condicional múltiple, junto con una variable que controla el flujo del programa después de cada operación. Si se establece en true, el programa sigue mostrando el menú; si se establece en false, el programa sale del bucle de menú y finaliza.
- El caso 1 es para mostrar la lista de los productos que ya se encuentran registrados en el inventario.
- El caso 2 es para insertar un nuevo producto en el inventario, y dentro de este caso se actualiza automáticamente el número de productos que está dentro del arreglo y nuevo máximo identificador de bebidas.
- El caso 3 es para buscar dentro del inventario el producto que el usuario quiera, para ello dentro de este caso se insertó la función de búsqueda binaria, por lo cual esto nos obliga a utilizar un algoritmo de ordenamiento.
- El caso 4 es para actualizar algún dato de los productos dentro del inventario,
- El caso 5 es para eliminar un producto del inventario, para lo cual se necesitó crear una función de eliminación, además que se necesita volver a llamar a la función de ordenamiento, ya que a la ausencia de un producto se deben de acomodar de nuevo los productos, y de ese modo la función de búsqueda binaria funcione.
- El caso 6 es para salir del menú, por lo tanto simplemente la variable continuarMenu la asigna como falsa, lo que indica que el usuario eligió salir del menú.
- En el caso de no ingresar uno de los posibles casos el programa llama a opcionInvalida(opcionLogin) para manejar la opción no válida.
- Llama a funciones como gotoxy, cout, retrasar y limpiar para manejar la salida de error y la interfaz de usuario.
- Guardar inventario: El programa permite a los usuarios guardar el estado actual del inventario en un archivo de Excel.

D. Diagrama de flujo



V. CRONOGRAMA DE ACTIVIDADES

SEMANAS	Semana 1		Semana 2		Semana 3	
	19/06/24	20/06/24	21/06/24	22/04/24	08/07/24	09/07/24
	Definir la estructura del sistema y las operaciones que se puedan realizar	Diseñar la interfaz del sistema	Implementar algoritmos de búsqueda y manejo de operaciones	Probar el sistema y asegurarnos de funciones correctamente	Realizar las correcciones pertinentes	Entregar el trabajo final
	Diseñar las opciones de inicio de sesión.		Realización del informe <ul style="list-style-type: none"> - Desarrollo de la solución. - Diagrama de flujo - Código fuente 	Mejorar el diseño del programa	Depurar errores y mejorar el sistema	Exponer el trabajo final

VI. CONCLUSIONES

- Un menú de opciones intuitivo y fácil de usar proporciona a los usuarios una forma clara y directa de interactuar con las funcionalidades del programa. La claridad en las opciones, la numeración y las instrucciones claras, pueden mejorar significativamente la experiencia del usuario.
- La interfaz gráfica desarrollada es amigable y visualmente atractiva, esta es una estrategia efectiva para mejorar la experiencia del usuario, aumentar la adopción del programa y destacar en el mercado. Al priorizar la usabilidad y la estética, se crea un programa que no solo funciona bien, sino que también se siente bien.

- La implementación de funcionalidades (Crear, Leer, Actualizar, Eliminar) es fundamental para permitir a los usuarios gestionar eficazmente los registros de la entidad seleccionada. Por lo que al desarrollar estas funcionalidades, es crucial utilizar estructuras de datos y algoritmos apropiados para garantizar la elección adecuada de estructuras de datos, mejorando significativamente el tiempo de acceso y procesamiento de los datos, especialmente en escenarios con grandes volúmenes de información. Asimismo, la implementación de algoritmos eficientes, como búsqueda binaria o ordenamiento rápido, puede optimizar el tiempo de ejecución de operaciones clave.
- Documentar detalladamente el proceso de desarrollo, incluyendo diagramas de flujo fue esencial para el proyecto ya que estos son especialmente útiles para visualizar la lógica y el flujo de control del programa, facilitando la identificación de posibles problemas o áreas de mejora.

VII. RECOMENDACIONES PARA EL USO DEL PROGRAMA

- Seguridad de las Contraseñas:
Asegúrese de que las contraseñas sean únicas y complejas para cada usuario. Nunca comparta su contraseña y mantenga un registro seguro de la misma, ya que el programa no permite cambios de contraseña.
- Regularidad en el Registro de Datos:
Mantenga el inventario actualizado registrando entradas y salidas de productos con regularidad. Esto asegurará que los datos del inventario sean precisos y útiles para la toma de decisiones.
- Revisiones Periódicas:

Realice revisiones periódicas del inventario para identificar y corregir posibles errores en los registros.

- Capacitación de Usuarios:

Capacite adecuadamente a los usuarios en el uso del programa, especialmente en las diferencias entre los permisos de administrador y empleado.

- Respaldo de Datos:

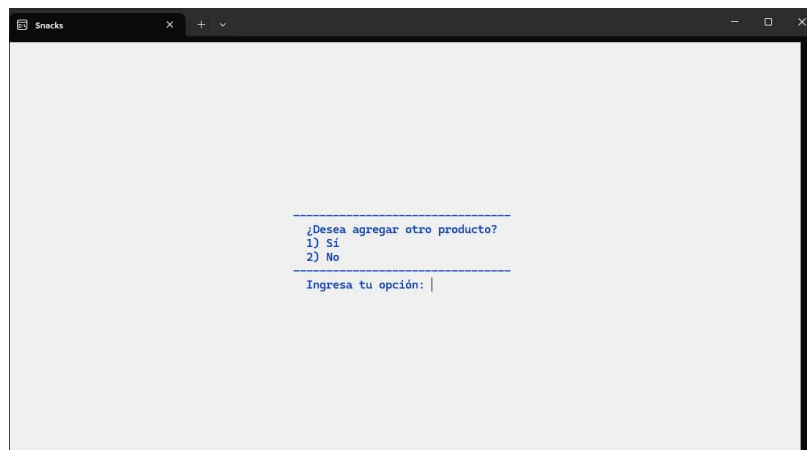
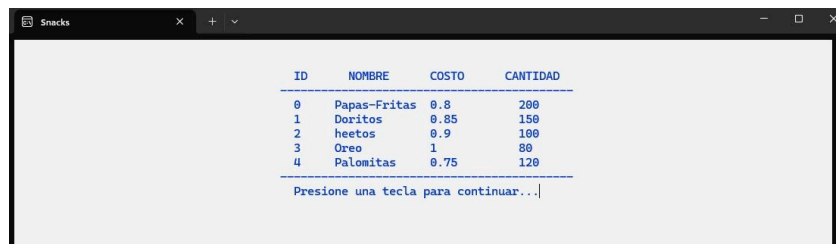
Realice copias de seguridad regulares del archivo Excel donde se guarda el inventario para evitar la pérdida de datos en caso de fallos del sistema.

Bibliografía:

- Gómez, E, Cañipa, P. (2023). Estructura de datos. (1ª Ed.). Alphaeditorial.
<https://www.alphaeditorialcloud.com/reader/estructuras-de-datos->
- TylerMSFT. (2024, 10 enero). *pair (Estructura)*. Microsoft Learn.
<https://learn.microsoft.com/es-es/cpp/standard-library/pair-structure?view=msvc-170>
- Garrido, A., & Sutil, M. A. (1998). *Programación en lenguajes estructurados*.
http://www.asuarseb.com/Archivos/PRJ_Tema6.pdf

VIII. Anexos

A. Imágenes del proyecto



IX. DÍPTICO:

CONCLUSIONES

El proyecto ofreció una experiencia completa del ciclo de desarrollo, desde la planificación hasta la implementación, destacando la importancia de cada fase. El uso de arreglos y registros fue efectivo para un inventario simple, pero se identificó la necesidad de optimizar el manejo de grandes volúmenes de datos en futuros proyectos, subrayando la importancia de elegir las estructuras de datos adecuadas según la complejidad del inventario.



SOLUCIÓN DE GESTIÓN DE INVENTARIOS AUTOMATIZADA PARA PEQUEÑAS EMPRESAS

CURSO: ESTRUCTURA DE DATOS

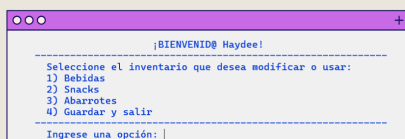
INTEGRANTES:
Mariela Ramos Loza
Marymar Calloticona
Fatima Yupa
Andre Carbajal



DESARROLLO DE LA SOLUCIÓN

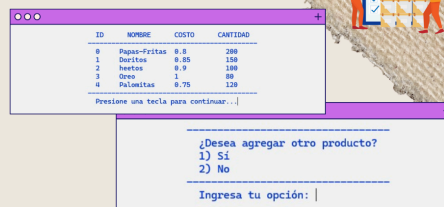


- Roles de usuario: El programa cuenta con dos roles de usuario:
 - Administradores: Tienen la capacidad de realizar todas las operaciones disponibles sobre los productos.
 - Empleados: Tienen acceso a un conjunto limitado de operaciones, como la visualización y búsqueda de productos.
- Clasificación de productos:
 - Bebidas.
 - Snacks.
 - Abarrotes.



- Gestión de productos: El sistema permite a los usuarios llevar a cabo diversas operaciones relacionadas con los productos, que incluyen:

- Visualización de la lista de productos
- Inserción de nuevos productos
- Búsqueda por ID de productos
- Actualización de la información del producto
- Eliminación de productos



- Además guardamos el inventario en un excel.

	A	B	C	D	E	F
1	Inventario de Sistema Integrado					
2						
3	Bebidas					
4	ID	nombre	costo	cantidad	ml	
5	0	Coca-Cola	1.5	100	500	
6	1	Pepsi	1.45	80	500	
7	2	Fanta	1.4	60	500	
8	3	Sprite	1.3	90	500	
9	4	Agua-Cielo	1	120	600	
10						

A. Video demo del proyecto

https://drive.google.com/file/d/1CviFZ5PYXYAJf_h3pT_dZUDxiJg8AFFN/view?usp=sharing