

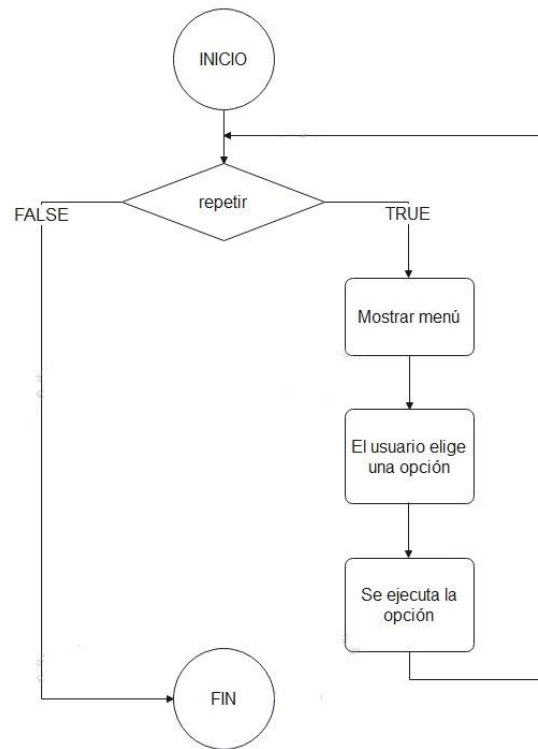
Descripción del problema.

El problema se origina como el segundo proyecto del portafolio del curso fundamentos de informática, para el primer ciclo 2022. Se trata de desarrollar una aplicación para ejercer una correcta administración de un supermercado, de manera que por medio de la aplicación se logre mantener el control de todos los productos. Algunos datos para tomar en cuenta para el desarrollo de este sistema son el código del producto, el nombre, el precio, la existencia, el peso, entre otros. Para el correcto funcionamiento de la aplicación es necesario organizar la información de todos los productos para así tener un registro.

1. Análisis de la solución del problema

El desarrollo del proyecto requiere utilizar diferentes herramientas para dar solución al siguiente problema: Es necesario crear una aplicación para organizar, añadir, eliminar y ejecutar algunas otras funciones para ciertos productos. Así mismo, ayudar con el manejo y la administración del supermercado. De manera que para lograr el propósito establecido se implementaron 2 clases; la primera tiene como nombre “Producto” y funciona para almacenar los datos de los productos, y la segunda, llamada “Supermercado” cumple la función de un supermercado, es decir, es la colección de todos los productos. Dentro de la clase “Supermercado” se implementaron los métodos necesarios para facilitar el manejo de toda la información. También fuera de las clases se utilizó una función para la implementación del menú, la cual le muestra al usuario todas las opciones disponibles, además se creó una función en la cual el usuario puede introducir su opción y esta le imprimirá un resultado en base a la opción escogida.

Elaboramos un diagrama de flujo para tener una idea clara de los pasos que se debían llevar a cabo para solucionar el problema.



2. Descripción de los aspectos más importantes a utilizar.

Atributos:

Clase Producto:

- *codigo*: Hilera de caracteres que identifica de forma única a cada producto.
- *nombre*: Nombre del producto.
- *precioBase*: Precio que paga el supermercado por el producto.
- *porcentajeDeGanancias*: Valor entre 0 y 1 que representa el porcentaje de ganancia que obtiene el supermercado por la venta del producto y permite calcular su precio de venta.
- *cantidadVendida*: Cantidad de unidades del producto que se han vendido.
- *existencia*: Cantidad de unidades del producto que están disponibles en el supermercado.
- *peso*: Peso de cada unidad del producto en kilogramos.
- *existenciaMinima*: cantidad mínima de unidades del producto que debe mantenerse en el supermercado.
- *precioDeVenta*: Precio que paga el cliente por un producto, se calcula multiplicando el precio base por el porcentaje de ganancias, todo esto sumado al precio base.

Clase Supermercado:

- *productos[]*: Almacena la información de todos los productos.

- *tamano*: Total de productos que se pueden ingresar. También se puede ver como el tamaño del supermercado.
- *cantidad*: Cantidad de productos ingresados.

Clases:

- *Producto*: Representa un producto.
- *Supermercado*: Representa un supermercado.

Funciones:

- *void menu(int opcion)*: Devuelve un string que contiene el menú de opciones.
- *void eligeOpcion(Supermercado &supermercado, int opcion, bool &repetir, int i)*: Es una sentencia switch que llama a un método de la clase Supermercado dependiendo de la opción elegida por el usuario.

Métodos:

Clase Producto:

Constructores:

- *Producto()*: Constructor por defecto de la clase Producto.
- *Producto(string codigo, string nombre, int precioBase, float porcentajeDeGanancias, int cantidadVendida, int existencia, float peso, int existenciaMinima)*: Constructor que inicializa los atributos de la clase producto con la información pasada por parámetro.

Destructor:

- *~Producto()*: Destructor de la clase producto.

Setters:

- *void setCodigo (string codigo)*: Esta función establece el valor del atributo codigo al valor del parámetro codigo.
- *void setNombre (string nombre)*: Esta función establece el valor del atributo nombre al valor del parámetro nombre.
- *void setPrecioBase (int precioBase)*: Esta función establece el valor del atributo precioBase al valor del parámetro precioBase.
- *void setPorcentajeDeGanancias (float porcentajeDeGanancias)*: Esta función establece el valor del atributo porcentajeDeGanancias al valor del parámetro porcentajeDeGanancias.
- *void setCantidadVendida (int cantidadVendida)*: Esta función establece el valor del atributo cantidadVendida al valor del parámetro cantidadVendida.
- *void setExistencia (int existencia)*: Esta función establece el valor del atributo existencia al valor del parámetro existencia.
- *void setPeso (float peso)*: Esta función establece el valor del atributo peso al valor del parámetro peso. El peso se representa en kilogramos.
- *void setExistenciaMinima (int existenciaMinima)*: Esta función establece el valor del atributo existenciaMinima al valor del parámetro existenciaMinima.

Getters:

- *string getCodigo()*: Devuelve el valor del atributo codigo.
- *string getNombre()*: Devuelve el valor del atributo nombre.
- *int getPrecioBase()*: Devuelve el valor del atributo precioBase.
- *float getPorcentajeDeGanancias()*: Devuelve el valor del atributo porcentajeDeGanancias.

- *int getCantidadVendida()*: Devuelve el valor del atributo cantidadVendida.
- *int getExistencia()*: Devuelve el valor del atributo existencia.
- *float getPeso()*: Devuelve el valor del atributo peso.
- *int getExistenciaMinima()*: Devuelve el valor del atributo existenciaMinima.
- *float getPrecioDeVenta()*: Devuelve el valor del atributo precioDeVenta.

Métodos:

- *string mostrarDatosDelProducto()*: Devuelve todos los atributos de un objeto de la clase Producto.

Clase Supermercado:

Constructores:

- *Supermercado()*: Constructor por defecto de la clase Supermercado.
- *Supermercado (int tamano)*: Constructor que inicializa los atributos de la clase Supermercado con la información pasada por parámetro

Destructor:

- *~Supermercado()*: Destructor de la clase Supermercado.

Setters:

- *void setTamano (int tamano)*: Esta función establece el valor del atributo tamano al valor del parámetro tamano.
- *void setCantidad (int cantidad)*: Esta función establece el valor del atributo cantidad al valor del parámetro cantidad.

Getters:

- *int getTamano()*: Devuelve el valor del atributo tamano.
- *int getCantidad()*: Devuelve el valor del atributo cantidad.

Métodos:

- *void ingresaProducto(int posicionProducto, Producto producto)*: Agrega un producto al array de productos. Recibe dos parámetros, el parámetro posicionProducto indica en cuál posición del array se debe de agregar, y el parámetro producto es un objeto con toda la información del producto.
- *void eliminarProductoPorCodigo(string codigo)*: Elimina un producto del array en base al código ingresado por el usuario.
- *Producto productoDeMayorValor()*: Devuelve un objeto, este objeto es el producto con mayor precio de venta.
- *Producto productoConMayorExistencia()*: Devuelve un objeto, este objeto es el producto con mayor existencia de venta.
- *void ordenarPorCodigo()*: Ordena los productos alfabéticamente en base a su código.
- *string mostrarTodosLosProductos()*: Devuelve un string con los datos de todos los productos.
- *int cantidadDeProductosBajosDeExistencia()*: Devuelve la cantidad de productos cuya existencia esté por debajo de la existencia mínima.
- *string mostrarProductosBajosDeExistencia()*: Devuelve un string con los datos de todos los productos cuya existencia esté por debajo de la existencia mínima.
- *float kilogramosVendidos()*: Devuelve la suma de la multiplicación entre peso de cada producto y la cantidad vendida de cada.

- *float kilogramosVendidosDeUnProducto (string codigo)*: Devuelve la suma de los kilogramos vendidos de un producto en específico, según su código.
- *string productosMasCostososQueUnProducto (string codigo)*: Devuelve el número de producto, nombre, código y precio de venta de los productos que son más costosos que el producto con el código que se paso por parámetro.
- *int cantidadDeUnidadesVendidasDeUnProducto (string codigo)*: Devuelve el número de unidades vendidas de un producto dado su codigo.
- *int costoDelInventario()*: Devuelve el costo del inventario, este se calcula sumando la multiplicación entre el precio base y la existencia de cada uno de los productos.
- *int costoDeLasVentas()*: Devuelve el costo de las ventas, este se calcula sumando la multiplicación entre el precio base y la cantidad vendida de cada uno de los productos.
- *int valorDeLasVentas()*: Devuelve la suma de los precios de venta multiplicados por la cantidad vendida de cada producto, es decir; calcula el valor total de las ventas.
- *int gananciaDelSupermercadoPorVentas()*: Devuelve la ganancia total del supermercado por las ventas realizadas, esta se calcula sumando
- *string tablaDeGraficos()*: Devuelve un string que contiene una tabla con el nombre de los primeros 5 productos y su cantidad de ventas, dicha cantidad se representa con asteriscos.
- *string graficoDeVentaDeLos5MasVendidos()*: Ordena los productos en base a la cantidad vendida de mayor a menor, y luego devuelve un gráfico con los cinco productos más vendidos.
- *string graficoDeVentaDeLos5MenosVendidos()*: Ordena los productos en base a la cantidad vendida de menor a mayor, y luego devuelve un gráfico con los cinco productos menos vendidos
- *float promedioDeLosPreciosVentas()*: Devuelve el promedio de los precios de venta, este se calcula sumando la multiplicación de la cantidad vendida y el precio de venta. Luego esto se divide por la cantidad de productos.
- *bool existeProducto (string codigo)*: Devuelve **true** si el producto cuyo código se pasó como parámetros existe, **false** si no.

Variables:

main():

- *int opcion*: Almacena la opción digita por el usuario, más tarde opción se utiliza para elegir una de las opciones del menú de opciones.
- *string codigo*: Se encarga de recibir el código de un producto introducido por el usuario, se suele utilizar como argumento de las distintas funciones.
- *bool repetir*: Almacena **true** o **false** en base a si el usuario desea seguir utilizando el menú de opciones.

eligeOpcion():

- *bool continuar*: Almacena **true** o **false** en base a si el usuario desea agregar otro producto.
- *string codigo*: Almacena el código de un producto, este código se pasa como un input y se utiliza como argumento para crear un objeto de la clase Producto y para utilizar distintos métodos de la clase Producto.

- *string nombre*: Almacena el nombre de un producto, este nombre se pasa como un input y se utiliza como argumento para crear un objeto de la clase Producto
- *int precioBase*: Almacena el precio base de un producto, este precio se pasa como un input y se utiliza como argumento para crear un objeto de la clase Producto
- *int cantidadVendida*: Almacena la cantidad vendida de un producto, esta cantidad se pasa como un input y se utiliza como argumento para crear un objeto de la clase Producto
- *int existencia*: Almacena la cantidad de unidades de un producto que hay en el supermercado(stock), esta cantidad se pasa como un input y se utiliza como argumento para crear un objeto de la clase Producto
- *int existenciaMinima*: Almacena el valor de la cantidad mínimas de unidades del producto que debe mantenerse en el supermercado, esta cantidad se pasa como un input y se utiliza como argumento para crear un objeto de la clase Producto
- *float porcentajeDeGanancias*: Almacena el porcentaje de ganancias que tiene un producto, esta cantidad se pasa como un input y se utiliza como argumento para crear un objeto de la clase Producto.
- *float peso*: Almacena el valor del peso de cada unidad de un producto, este valor se pasa como un input, se suele utilizar como argumento de algunos métodos de la clase Producto.

Producto:

- *bool esVacio*: Se utiliza para comprobar si hay al menos un producto que cumpla con ese criterio.

3. Descripción de datos de prueba.

El programa se probó con una gran variedad de combinaciones, desde números negativos hasta números mayores a 100000. Es importante recalcar que cuando algún valor es un número negativo el programa funciona de la misma manera.

4. Limitaciones del programa.

Cuando se digita una cadena de texto en una variable que solo admite enteros o flotantes el programa entra en un bucle.

Además no se evalúan los datos introducidos por el usuario.

5. Observaciones generales.

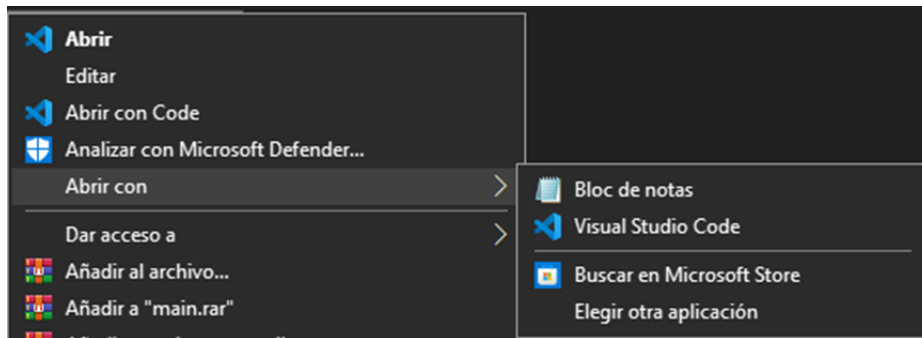
Al inicio del programa se incluye la biblioteca `sstream`, con el fin de poder tratar un string como un stream

6. Manual de usuario.

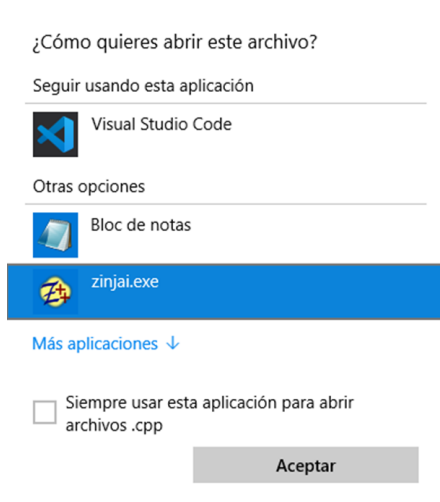
Descargar e instalar [Zinjai](#) o su entorno de desarrollo preferido, en este caso trabajaremos con [Zinjai](#).


Descargar el archivo [supermercado.cpp](#).

Buscamos el archivo recientemente descargado, le damos **click derecho >> abrir con >> elegir otra aplicación**.



Buscamos la aplicación Zinjai y le damos doble click izquierdo.



Dentro de la aplicación Zinjai ejecutamos nuestro código presionando F9 o con el botón de Save, Compile and Run .

Una vez se haya ejecutado nuestra aplicación debemos introducir la opción deseada (número entre 1-20).

Por último, se siguen las instrucciones de la aplicación.