INFORME DE PROGRAMACIÓN I.

(Desarrollo de un sistema de gestión de inventario en C)

Autores:

Macas Pachar Amelio.

Aimacaña Herrera Martin.

Palma Zambrano Alex.

Institución:

Universidad de las Américas.

Curso:

Ingeniería de Software

ISWZ1102 - Programación I

Docente:

Armas, Eddy Mauricio

Fecha de entrega:

02 de octubre del 2025.

1. Introducción

En el siguiente informe se realizo con la finalidad de dar a conocer los conocimientos que emos adquiridos en la materia de “Algoritmos”, donde se demostrara el mediante el lenguaje C. En si el ejercicio planteado consiste en diseñar un sistema para un local pequeño el cual nos va a permitir gestionar la venta de un solo producto dentro de la tienda y este sistema estará equipado para que ofrezca servicios como registro, cambio, venta, reabastecimiento, y consulta de la información.

Este programa nos resulta muy útil hoy en día para poder gestionar las cosas mas eficazmente ya sea por su automatización ya que todo se hace de forma automática. También se puede notar que el desarrollo del programa nos ayuda a retroalimentar nuestros antiguos conocimientos para poderlos ocupar en un futuro no tan distante

2. Implementación del Código

Para la resolución del problema, se planteo el algoritmo usando una estructura en C “struct” para el producto y atributos como lo son el: (ID, nombre, stock, precio unitario y la ganancia acumulada). Siendo que este es un sistema fácil de dar mantenimiento a todos sus funciones y atributos.

Entre los aspectos técnicos implementados destacan:

Validación de entradas: Se incluyo funciones que aseguran que el valor ingresado únicamente sea datos válidos(positivos, enteros).

Estructuras de control: se emplearon if y switch para verificar restricciones y gestionar las opciones del menú.

Cálculos matemáticos: se implementó la lógica para calcular descuentos, ganancias acumuladas y actualización de stock después de cada venta.

Iteraciones: mediante un bucle while se mantiene activo el programa hasta que el usuario decida salir.

A continuación, se muestran fragmentos representativos del código desarrollado (el programa completo se encuentra en el repositorio de GitHub):

Imagen del código

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define NAME_MAX_LEN 100

typedef struct {
    int id;
    char nombre[NAME_MAX_LEN];
    long stock;
    double precio_unitario;
    double ganancias;
} Producto;

/* --- Utilidades de entrada segura --- */
int leer_linea(char *buf, size_t n) {
    if (!fgets(buf, (int)n, stdin)) return 0;
    size_t len = strlen(buf);
    if (len && buf[len-1] == '\n') buf[len-1] = '\0';
    return 1;
}

int leer_entero(const char *prompt, long *out) {
    char buf[256];
    char *endptr;
    while (1) {
        printf("%s", prompt);
        if (!leer_linea(buf, sizeof buf)) return 0;
        if (buf[0] == '\0') { puts("Entrada vacia. Intente nuevamente."); continue; }
        long val = strtol(buf, &endptr, 10);
        if (*endptr == '\0') { *out = val; return 1; }
        puts("Entrada invalida. Ingrese un numero entero.");
    }
}

int leer_real(const char *prompt, double *out) {
    char buf[256];
    char *endptr;
    while (1) {
        printf("%s", prompt);
```

```
        if (!leer_linea(buf, sizeof buf)) return 0;
        if (buf[0] == '\0') { puts("Entrada vacia. Intente nuevamente."); continue; }
        double val = strtod(buf, &endptr);
        if (*endptr == '\0') { *out = val; return 1; }
        puts("Entrada invalida. Ingrese un numero (usar punto decimal).");
    }
}

/* --- Funciones del negocio --- */

void registrar_producto(Producto *p) {
    long idTmp;
    double precio;
    long stock;
    char nombre[NAME_MAX_LEN];

    puts("\n=== Registro/Actualizacion de Producto ===");
    leer_entero("ID (entero): ", &idTmp);
    printf("Nombre (max %d chars): ", NAME_MAX_LEN-1);
    leer_linea(nombre, sizeof nombre);
    leer_entero("Cantidad en stock (entero >= 0): ", &stock);
    leer_real("Precio unitario (>= 0): ", &precio);

    if (stock < 0) stock = 0;
    if (precio < 0) precio = 0.0;
    p->id = (int)idTmp;
    strncpy(p->nombre, nombre, NAME_MAX_LEN-1);
    p->nombre[NAME_MAX_LEN-1] = '\0';
    p->stock = stock;
    p->precio_unitario = precio;
    // ganancias no se toca en actualizacion
    puts("Producto registrado/actualizado correctamente.\n");
}

void mostrar_producto(const Producto *p) {
    puts("\n=== Informacion del producto ===");
    printf("ID: %d\n", p->id);
    printf("Nombre: %s\n", p->nombre);
    printf("Stock: %ld\n", p->stock);
    printf("Precio unitario: $%.2f\n", p->precio_unitario);
    printf("Ganancias acumuladas: $%.2f\n\n", p->ganancias);
}

void vender(Producto *p) {
    if (p->precio_unitario <= 0) {
        puts("Primero registre el producto con precio valido.");
        return;
    }
    long unidades;
    double descuento_pct = 0.0;

    puts("\n=== Venta ===");
    if (!leer_entero("Unidades a vender (>= 1): ", &unidades)) return;
    if (unidades <= 0) { puts("Las unidades deben ser positivas."); return; }
```

```c
    if (unidades > p->stock) {
        printf("No hay stock suficiente. Stock disponible: %ld\n", p->stock);
        return;
    }

    leer_real("Descuento (%) para esta venta (0-100): ", &descuento_pct);
    if (descuento_pct < 0) descuento_pct = 0;
    if (descuento_pct > 100) descuento_pct = 100;

    double precio_bruto = unidades * p->precio_unitario;
    double descuento = precio_bruto * (descuento_pct / 100.0);
    double total = precio_bruto - descuento;

    p->stock -= unidades;
    p->ganancias += total;

    printf("Venta realizada: %ld unidad(es) de '%s'\n", unidades, p->nombre);
    printf("Precio bruto: $%.2f | Descuento: $%.2f | Total: $%.2f\n", precio_bruto, descuento, total);
    printf("Stock restante: %ld\n\n", p->stock);
}

void reabastecer(Producto *p) {
    long unidades;
    puts("\n=== Reabastecer ===");
    if (!leer_entero("Unidades a agregar (>= 1): ", &unidades)) return;
    if (unidades <= 0) { puts("Las unidades deben ser positivas."); return; }
    if (unidades > 1000000000L) { puts("Cantidad demasiado grande."); return; }
    p->stock += unidades;
    printf("Reabastecimiento exitoso. Stock actual: %ld\n\n", p->stock);
}

void menu() {
    puts("========== MENU ==========");
    puts("1. Registrar/Actualizar producto");
    puts("3. Reabastecer");
    puts("4. Consultar informacion");
    puts("5. Mostrar ganancias");
    puts("0. Salir");
    puts("==========================");
}

int main(void) {
    Producto producto = {0};
    producto.ganancias = 0.0;

    int opcion;
    char buf[256];

    puts("Sistema de gestion de un unico producto (C)");
    while (1) {
        menu();
        printf("Opcion: ");
        if (!leer_linea(buf, sizeof buf)) break;
        if (sscanf(buf, "%d", &opcion) != 1) { puts("Opcion invalida."); continue; }
        switch (opcion) {
            case 1: registrar_producto(&producto); break;
            case 2: vender(&producto); break;
            case 3: reabastecer(&producto); break;
            case 4: mostrar_producto(&producto); break;
            case 5: printf("Ganancias acumuladas: $%.2f\n\n", producto.ganancias); break;
```

```
            case 0: puts("Saliendo..."); return 0;
            default: puts("Opcion invalida."); break;
        }
    }
    return 0;
}
```

Enlace al repositorio GitHub:

(Aquí va el enlace)

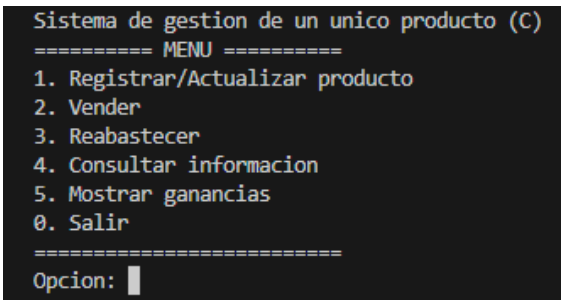
3. Validación y Pruebas

Para comprobar el correcto funcionamiento del programa, se realizaron distintas pruebas de validación:

- **Venta con stock insuficiente:** el programa bloquea la operación e informa al usuario la cantidad disponible.
- **Reabastecimiento:** al ingresar nuevas unidades, el stock aumenta correctamente y se refleja en las consultas posteriores.
- **Cálculo de ganancias:** cada vez que se realiza una venta, el sistema descuenta el stock, aplica el descuento (si corresponde) y actualiza el total acumulado de ganancias.
- **Manejo de entradas inválidas:** se verificó que el sistema rechaza números negativos, valores excesivamente grandes y datos no numéricos.

Imágenes de la ejecución del código

Ejecución del programa prueba 1



```
========== MENU ==========
1. Registrar/Actualizar producto
2. Vender
3. Reabastecer
4. Consultar informacion
5. Mostrar ganancias
0. Salir
=========================
Opcion: 1
```

```
Opcion: 1

=== Registro/Actualizacion de Producto ===
ID (entero): 20070412
Nombre (max 99 chars): RTX 3016 NVIDIA 8GB
Cantidad en stock (entero >= 0): 20
Precio unitario (>= 0): 400
Producto registrado/actualizado correctamente.
```

Ejecución del programa prueba 2

```
  Sistema de gestion de un unico producto (C)
  ========== MENU ==========
  1. Registrar/Actualizar producto
  2. Vender
  3. Reabastecer
  4. Consultar informacion
  5. Mostrar ganancias
  0. Salir
  =========================
  Opcion:
```

```
========== MENU ==========
1. Registrar/Actualizar producto
2. Vender
3. Reabastecer
4. Consultar informacion
5. Mostrar ganancias
0. Salir
=========================
Opcion: 2
```

```
==========================
Opcion: 2

=== Venta ===
Unidades a vender (>= 1): 10
Descuento (%) para esta venta (0-100): 2
Venta realizada: 10 unidad(es) de 'RTX 3016 NVIDIA 8GB'
Precio bruto: $4000.00 | Descuento: $80.00 | Total: $3920.00
Stock restante: 10
```

Ejecución del programa prueba 3

```
  Sistema de gestion de un unico producto (C)
  ========== MENU ==========
  1. Registrar/Actualizar producto
  2. Vender
  3. Reabastecer
  4. Consultar informacion
  5. Mostrar ganancias
  0. Salir
  ==========================
  Opcion:
```

```
========== MENU ==========
1. Registrar/Actualizar producto
2. Vender
3. Reabastecer
4. Consultar informacion
5. Mostrar ganancias
0. Salir
==========================
Opcion: 3
```

```
==========================
Opcion: 3

=== Reabastecer ===
Unidades a agregar (>= 1): 25
Reabastecimiento exitoso. Stock actual: 35
```

Ejecución del programa prueba 4

```
Sistema de gestion de un unico producto (C)
========== MENU ==========
1. Registrar/Actualizar producto
2. Vender
3. Reabastecer
4. Consultar informacion
5. Mostrar ganancias
0. Salir
=========================
Opcion:
```

```
========== MENU ==========
1. Registrar/Actualizar producto
2. Vender
3. Reabastecer
4. Consultar informacion
5. Mostrar ganancias
0. Salir
=========================
Opcion: 4

=== Informacion del producto ===
ID: 20070412
Nombre: RTX 3016 NVIDIA 8GB
Stock: 35
Precio unitario: $400.00
Ganancias acumuladas: $4661.00
```

Ejecución del programa prueba 5

```
Sistema de gestion de un unico producto (C)
========== MENU ==========
1. Registrar/Actualizar producto
2. Vender
3. Reabastecer
4. Consultar informacion
5. Mostrar ganancias
0. Salir
=========================
Opcion:
```

```
========== MENU ==========
1. Registrar/Actualizar producto
2. Vender
3. Reabastecer
4. Consultar informacion
5. Mostrar ganancias
0. Salir
=========================
Opcion: 5
Ganancias acumuladas: $4661.00
```

Ejecución del programa prueba 5

```
Sistema de gestion de un unico producto (C)
========== MENU ==========
1. Registrar/Actualizar producto
2. Vender
3. Reabastecer
4. Consultar informacion
5. Mostrar ganancias
0. Salir
=========================
Opcion:
```

```
========== MENU ==========
1. Registrar/Actualizar producto
2. Vender
3. Reabastecer
4. Consultar informacion
5. Mostrar ganancias
0. Salir
=========================
Opcion: 0
Saliendo...
PS C:\Users\AME>
PS C:\Users\AME>
PS C:\Users\AME>
```

4. Conclusiones

El desarrollo de este programa nos a permitido reforzar fundamentos en la programación en C, ya sea el manejo de estructuras, funciones, validaciones y estructuras de control.

Entre los principales aprendizajes destacan:

- La importancia de validar adecuadamente las entradas
- La utilidad de dividir en funciones específicas, lo cual facilita la comprensión y mantenimiento del programa.
- La aplicación práctica de estructuras como struct para modelar

Los principales retos enfrentados fueron:

- Diseñar la lógica matemática para calcular correctamente las ganancias y aplicar descuentos en cada venta.
- Mantener el programa organizado y legible, utilizando menús y mensajes claros para el usuario.

En conclusión, este ejercicio no solo sirvió para cumplir con los requerimientos de las instrucciones, sino también como una práctica valiosa para desarrollar habilidades de resolución de problemas mediante la programación.