Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria de Ingeniería y Tecnologías Avanzadas

Práctica 1

Profesor

Iliac Huerta Trujillo

Unidad de Aprendizaje

Programación Orientada a Objetos

Presenta

Ramírez López Emilio	2022640230
Sánchez Díaz Nadya	2022640236
Velázquez Osorio Samara Ishtar	2022640188

20 de septiembre de 2022

Índice

Índice de figuras

1. Planteamiento del problema

Un almacén de pedidos por correo cuenta con 5 productos distintos en inventario identificados como producto 1, producto 2, producto 3, producto 4 y producto 5. De los cuales sus respectivos precios son $2.98, $4.50, $9.98, $4.49 y $5.58.

Se busca un método eficiente para calcular la cantidad de productos vendidos y el total de ingresos por venta y al final del día, al cierre.

2. Justificación

Un programa de este tipo puede ser de mucha utilidad como base para el sistema de diferentes negocios y así tener un control y registro de las ventas realizadas día a día, haciendo de este un proceso más eficiente al llevarlo a cabo de forma automática.

Además, resulta un ejercicio útil para familiarizarnos con estructuras tales como do-while y switch.

3. Propuesta de solución

Creamos un programa que funcione como registro de las compras que se realizan por medio de los datos que se reciben. Cuando el usuario señale que ya no requiere de más productos, se le imprimirá un recibo que marque cuántos productos se compraron y cuánto es el total de la venta. Al finalizar las ventas, estas se irán sumando para que, cuando se solicite el cierre de ventas del día, se imprima el registro de la cantidad total vendida de cada producto y el valor total de la venta del día.

El programa lee una serie de números que se introducen consecutivamente para poder conocer:

1. El número del producto
2. La cantidad del producto

4. Análisis y diseño

Al analizar el problema podemos ver que necesitamos usar la estructura switch para controlar el precio total de cada producto que se vende y también el uso de do-while para realizar la primera venta y seguir repitiendo hasta que se nos pida cerrar la cuenta actual y también la cuenta al final del día. Cada case en el switch será un producto diferente y nos permitirá calcular la suma del costo total de la venta.

A continuación se presentan diagramas que ayudarán a comprender de manera gráfica nuestra propuesta de solución.

1. *Diagrama de actividades:* Es un tipo de diagrama dentro del lenguaje unificado de modelado (UML). El cual nos permitirá visualizar de manera general el flujo de actividades dentro de nuestro programa.
2. *Diagrama de flujo:* Representa la esquematización gráfica de un algoritmo, el cual muestra los pasos a seguir para la solución de un problema.

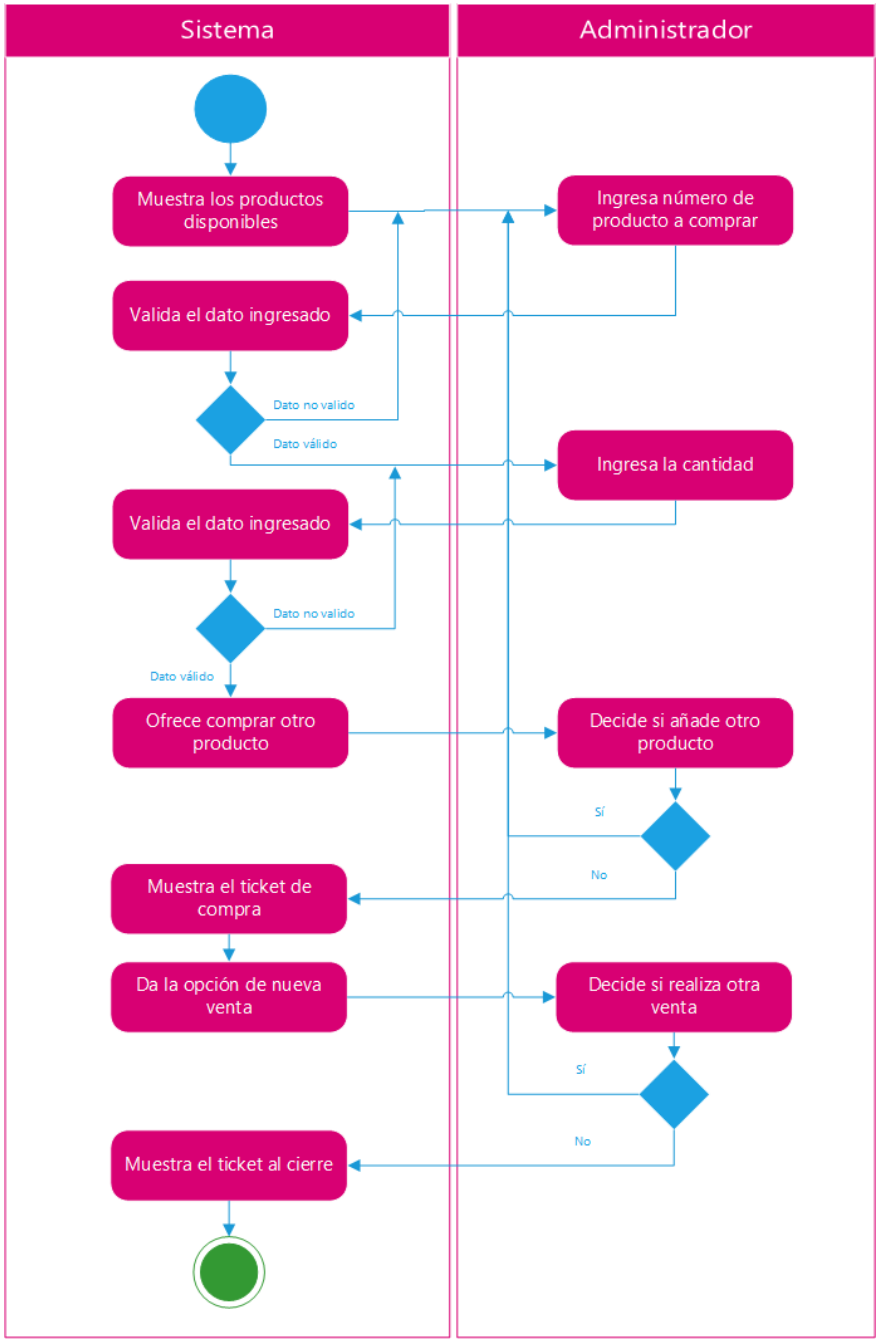

Figura 1: Diagrama de actividades

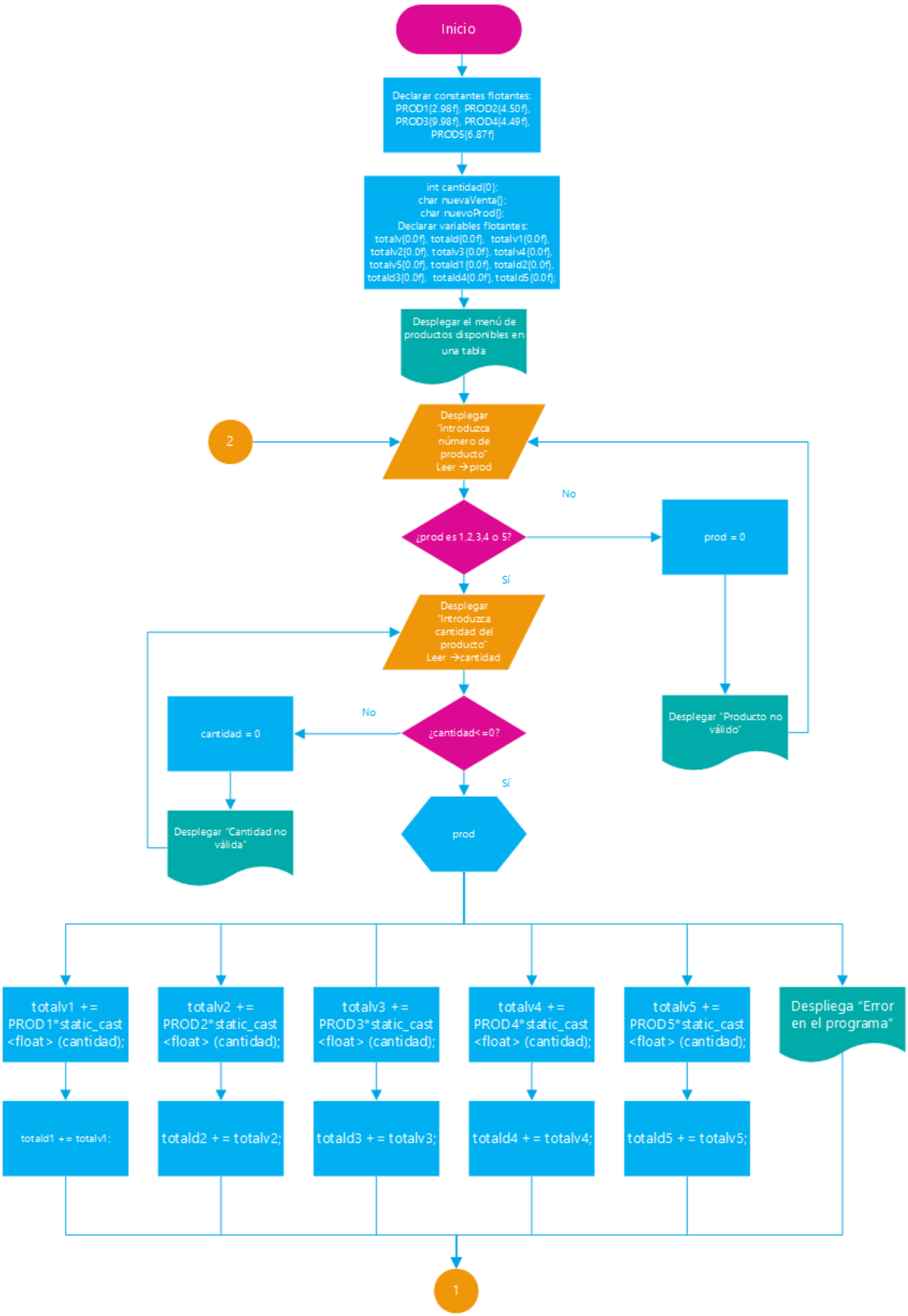

Figura 2: Diagrama de flujo

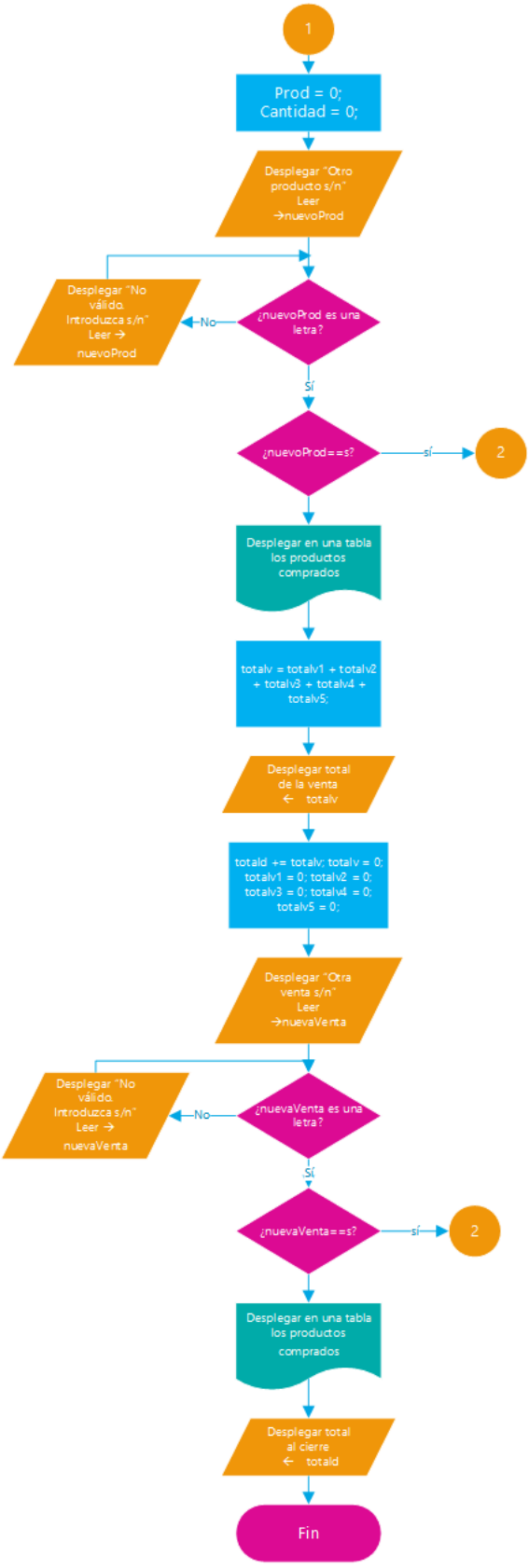

Figura 3: Diagrama de flujo *continuación*

5. Desarrollo e implementación

5.1. Desarrollo del programa

En el programa, comenzamos declarando como constantes de tipo flotante los precios de cada producto.

```
23      const float PROD1{2.98f};
24      const float PROD2{4.50f};
25      const float PROD3{9.98f};
26      const float PROD4{4.49f};
27      const float PROD5{6.87f};
```

Figura 4: Constantes

Seguida de variables de tipo entero para almacenar el número de producto y su cantidad, variables de tipo caracter para dar la opción de continuar con las ventas o no, y variables de tipo flotante para ir almacenando el total de ingresos de cada producto y total por venta y día.

```
29      int prod{0};        /**<Guarda el número de producto escogido por el usuario*/
30      int cantidad{0};    /**<Guarda la cantidad de producto ingresada por el usuario*/
31      char nuevaVenta{};    /**<Dependiendo de lo que almacene, se registrarán más ventas*/
32      char nuevoProd{};     /**<Dependiendo de lo que almacene, se registrarán más productos*/
33      float totalv{0.0f};   /**<Va almacenando los ingresos totales por venta*/
34      float totald{0.0f};   /**<Va almacenando los ingresos totales al cierre*/
35
36     /**
37      * @brief <Van almacenando los ingresos de cada uno de los productos por venta (el numero al final de la variable corresponde al numero de producto)
38      */
39      float totalv1{0.0f};
40      float totalv2{0.0f};
41      float totalv3{0.0f};
42      float totalv4{0.0f};
43      float totalv5{0.0f};
44
45     /**
46      * @brief <Van almacenando los ingresos de cada uno de los productos por día (el numero al final de la variable corresponde al numero de producto)
47      */
48      float totald1{0.0f};
49      float totald2{0.0f};
50      float totald3{0.0f};
51      float totald4{0.0f};
52      float totald5{0.0f};
```

Figura 5: Variables

Decidimos agregar una tabla donde se muestren los productos disponibles y su precio correspondiente para facilitar al usuario el uso del programa.

```
60      cout << setw(13)<< " Producto 1" << setw(18) << "$ 2.98 \n"
61           << setw(13)<< " Producto 2" << setw(18) << "$ 4.50 \n"
62           << setw(13)<< " Producto 3" << setw(18) << "$ 9.98 \n"
63           << setw(13)<< " Producto 4" << setw(18) << "$ 4.49 \n"
64           << setw(13)<< " Producto 5" << setw(18) << "$ 6.87 \n" << endl;
```

Figura 6: Inventario

Usamos un ciclo do-while para realizar al menos una venta y tener la posibilidad de continuar con otra siempre que el usuario así lo desee (que ingrese una 's'), de otra forma se dará por finalizado el día.

Dentro de este ciclo tenemos otro do-while que permitirá seleccionar un producto y la cantidad deseada, dando la opción a seguir añadiendo productos siempre que el usuario ingrese una 's', de lo contrario, se cerraría la venta.

Para esto, necesitamos que el usuario ingrese un número valido con el que se identifica un producto existente, para ello usamos un ciclo while, en el que se solicita al usuario ingresar el número de producto, con la condición de que prod sea igual a cero, debido a que prod se inicializó con cero, se ejecutará inicialmente y prod solo cambiará si el usuario introduce un número de tipo entero. Dentro de este while agregamos un if para asegurarse de que en caso de que se ingrese un número entero, este sea únicamente 1, 2, 3, 4 o 5.

```
74          do{
75              /**
76              * @brief while que valida que el dato ingresado sea un número
77              */
78              while(prod == 0){
79                  cout << "\n Introduzca el numero del producto: ";
80
81                  cin >> prod;
82
83                      /**
84                      * @brief if que valida la existencia del producto
85                      */
86                  if(prod == 1||prod == 2||prod == 3||prod == 4||prod == 5){
87                      cin.ignore(numeric_limits<streamsize>::max(), '\n');
88                  }else{
89                      cin.clear();
90                      cin.ignore(numeric_limits<streamsize>::max(), '\n');
91                      prod = 0;
92                      cout << " Producto no valido" << endl;
93                  }
94              }
```

Figura 7: Validación de producto

Hicimos una validación similar al momento de solicitar la cantidad del producto seleccionado, con la única diferencia de que la condición del if (que muestra el mensaje de error) es que la cantidad sea un número menor o igual a cero.

```
98              while(cantidad == 0){
99                  cout << "\n Introduzca la cantidad del producto: ";
100                 cin >> cantidad;
101                     /**
102                     * @brief if que valida que el dato sea un numero positivo
103                     */
104                 if(cantidad <= 0){
105                     cin.clear();
106                     cin.ignore(numeric_limits<streamsize>::max(), '\n');
107                     cantidad = 0;
108                     cout << " Cantidad no valida" << endl;
109                 }
110             }
```

Figura 8: Validación de cantidad

Una vez validado el producto y su cantidad, usamos un switch en el que cada producto representa un caso distinto, y según sea el caso hará lo siguiente:

1. Multiplicará el precio del producto por la cantidad solicitada y se acumulará en el total por venta de dicho producto.

2. Este total por venta se irá acumulando en otra variable que representa el ingreso total del producto al cierre.

```
117             switch(prod){
118             case 1:
119                 totalv1 += PROD1*static_cast <float> (cantidad);
120                 totald1 += totalv1;
121                 break;
122             case 2:
123                 totalv2 += PROD2*static_cast <float> (cantidad);
124                 totald2 += totalv2;
125                 break;
126             case 3:
127                 totalv3 += PROD3*static_cast <float> (cantidad);
128                 totald3 += totalv3;
129                 break;
130             case 4:
131                 totalv4 += PROD4*static_cast <float> (cantidad);
132                 totald4 += totalv4;
133                 break;
134             case 5:
135                 totalv5 += PROD5*static_cast <float> (cantidad);
136                 totald5 += totalv5;
137                 break;
138             default:
139                 cout << "\n Error en el programa" << endl;
140                 break;
141             }
```

Figura 9: Estructura switch

Regresamos a prod y cantidad a su valor inicial (0) para no afectar el funcionamiento de ciclos futuros en caso de que se decida agregar un nuevo producto a la venta, y para ello solicitamos al usuario ingrese 's' o 'n' según sea el caso, validando con isalpha que sea un caracter y usando tolower para que no haya diferencia entre mayúsculas y minúsuculas.

```
143             prod = 0;
144             cantidad = 0;
145
146             /**
147             * @brief Pregunta si desea agregagar otro producto mediante s/n y válida que el dato sea una letra
148             */
149
150             cout << "\n\t Otro producto s/n: ";
151             cin.ignore(numeric_limits<streamsize>::max(), '\n');
152             cin >> nuevoProd;
153             while(!isalpha(nuevoProd)){
154                 cout << "\n\tNo valido. Introduzca s/n: ";
155                 cin >> nuevoProd;
156             }
157             cin.ignore();
158         }while(tolower(nuevoProd) == 's');
```

Figura 10: Condición del do-while interno

Si se eligió terminar la venta, se despliega en pantalla una tabla a modo de ticket en el que se muestra el producto vendido, la cantidad, el monto de cada producto y abajo el monto total de la venta.

```
163         cout << "\n     -------------------------" << endl;
164         cout << setw(13)<< "Producto" << setw(10) << "Cantidad"<< setw(9) << "Total" << endl;
165         cout << setw(9)<< "1" << setw(10) << totalv1/PROD1 << setw(10) << "$" << totalv1 << endl;
166         cout << setw(9)<< "2" << setw(10) << totalv2/PROD2 << setw(10) << "$" << totalv2 << endl;
167         cout << setw(9)<< "3" << setw(10) << totalv3/PROD3 << setw(10) << "$" << totalv3 << endl;
168         cout << setw(9)<< "4" << setw(10) << totalv4/PROD4 << setw(10) << "$" << totalv4 << endl;
169         cout << setw(9)<< "5" << setw(10) << totalv5/PROD5 << setw(10) << "$" << totalv5 << endl;
170
171             totalv = totalv1 + totalv2 + totalv3 + totalv4 + totalv5;
172
173         cout << "\n ::::: Total de la venta: $" << totalv << " :::::" << endl;
174         totald += totalv;
175         totalv = 0;
176         totalv1 = 0; totalv2 = 0; totalv3 = 0; totalv4 = 0; totalv5 = 0;
```

Figura 11: Ticket de venta

Usando la misma lógica, se da la opción al usuario de realizar una nueva venta y se valida el dato ingresado de la misma forma. De no haber elegido 's' salimos del ciclo do-while y se imprime una tabla similar a la de cada venta con la diferencia de mostrar el monto acumulado en todo el día.

```
181         cout << "\n\tNueva venta s/n: ";
182         cin >> nuevaVenta;
183         while(!isalpha(nuevaVenta)){
184           cout << "\n\tNo valido. Introduzca s/n: ";
185           cin >> nuevaVenta;
186         }
187         cin.ignore();
188
189     }while(tolower(nuevaVenta) == 's');
190
191     /**
192     * @brief Imprime el ticket de las ventas al cierre
193     */
194     cout << "\n     -------------------------" << endl;
195     cout << setw(13)<< "Producto" << setw(10) << "Cantidad"<< setw(9) << "Total" << endl;
196     cout << setw(9)<< "1" << setw(10) << totald1/PROD1 << setw(10) << "$" << totald1 << endl;
197     cout << setw(9)<< "2" << setw(10) << totald2/PROD2 << setw(10) << "$" << totald2 << endl;
198     cout << setw(9)<< "3" << setw(10) << totald3/PROD3 << setw(10) << "$" << totald3 << endl;
199     cout << setw(9)<< "4" << setw(10) << totald4/PROD4 << setw(10) << "$" << totald4 << endl;
200     cout << setw(9)<< "5" << setw(10) << totald5/PROD5 << setw(10) << "$" << totald5 << endl;
201     cout << "     -------------------------" << endl;
202
203     cout << "\n <<<<< Venta total del dia: $" << totald << " >>>>>" << endl;
204
205     return 0;
206 }
```

Figura 12: Condición del do-while externo y ticket al cierre

5.2. Implementación

Incluimos un ejemplo del funcionamiento del programa. El usuario hará 3 ventas:

1. En la primer venta se comprarán 2 productos:
   - *a*) 26 unidades del producto 4.
   - *b*) 31 unidades del producto 1.
2. En la segunda venta se comprarán 4 productos:
   - *a*) 96 unidades del producto 5.

b) 300 unidades del producto 2.

c) 42 unidades del producto 4.

d) 50 unidades del producto 3.

3. En la tercera venta se comprará 1 producto:

   a) 11 unidades del producto 3.

```
:::VENTA DE PRODUCTOS:::
Almacen de pedidos por correo

Contamos con los siguientes productos en nuestro inventario:

  Producto 1          $ 2.98
  Producto 2          $ 4.50
  Producto 3          $ 9.98
  Producto 4          $ 4.49
  Producto 5          $ 6.87


Introduzca el numero del producto: 4

Introduzca la cantidad del producto: 26

        Otro producto s/n: s

Introduzca el numero del producto: 1

Introduzca la cantidad del producto: 31

        Otro producto s/n: n

    -------------------------
    Producto  Cantidad    Total
       1        31         $92.38
       2         0         $0
       3         0         $0
       4        26         $116.74
       5         0         $0

::::: Total de la venta: $209.12 :::::

       Nueva venta s/n: s
```

Figura 13: Ejemplo venta 1

```
Introduzca el numero del producto: 5

Introduzca la cantidad del producto: 96

        Otro producto s/n: s

Introduzca el numero del producto: 2

Introduzca la cantidad del producto: 300

        Otro producto s/n: s

Introduzca el numero del producto: 4

Introduzca la cantidad del producto: 42

        Otro producto s/n: s

Introduzca el numero del producto: 3

Introduzca la cantidad del producto: 50

        Otro producto s/n: n

    -------------------------
    Producto  Cantidad    Total
       1         0         $0
       2       300         $1350
       3        50         $499
       4        42         $188.58
       5        96         $659.52

::::: Total de la venta: $2697.1 :::::

       Nueva venta s/n: s
```

Figura 14: Ejemplo venta 2

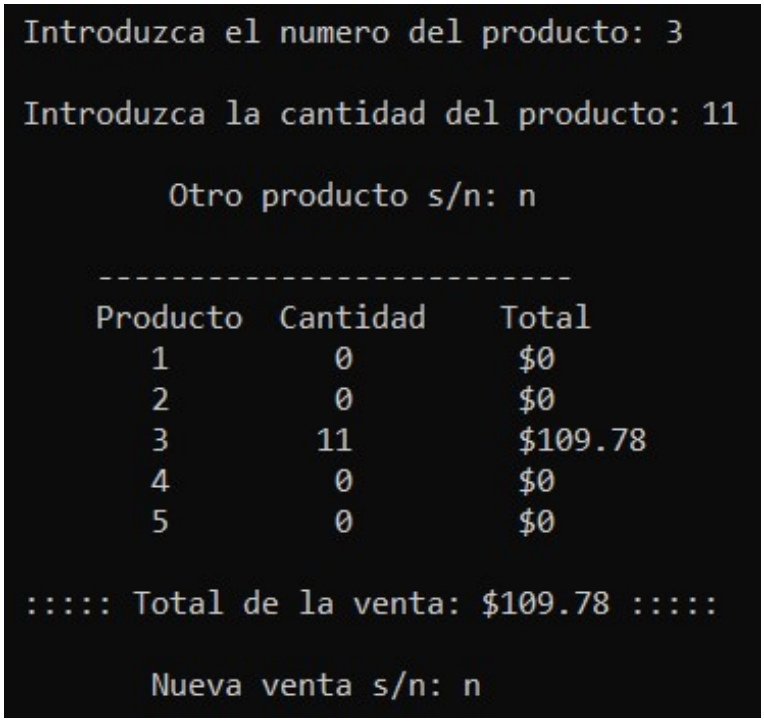

Figura 15: Ejemplo venta 3

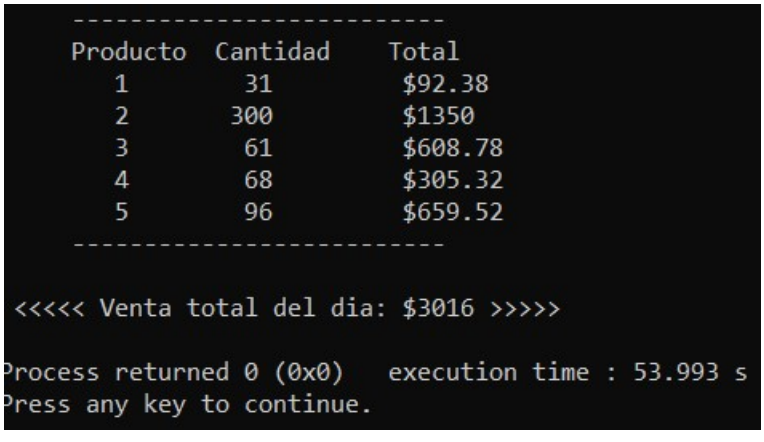

Figura 16: Ejemplo cierre

6. Conclusiones

En la realización de la práctica se trabajó con uno de los usos prácticos que se le puede dar a la sentencia switch asignando cada caso a un producto. Además de comprobar la utilidad del ciclo do-while al realizar iteraciones hasta que se indique que ya no se va a comprar más producto en una venta o que se finalizaron las ventas del día. De igual forma, en la validación de datos se implementaron verificadores del flujo de entrada para evitar problemas de iteraciones infinitas sin control.