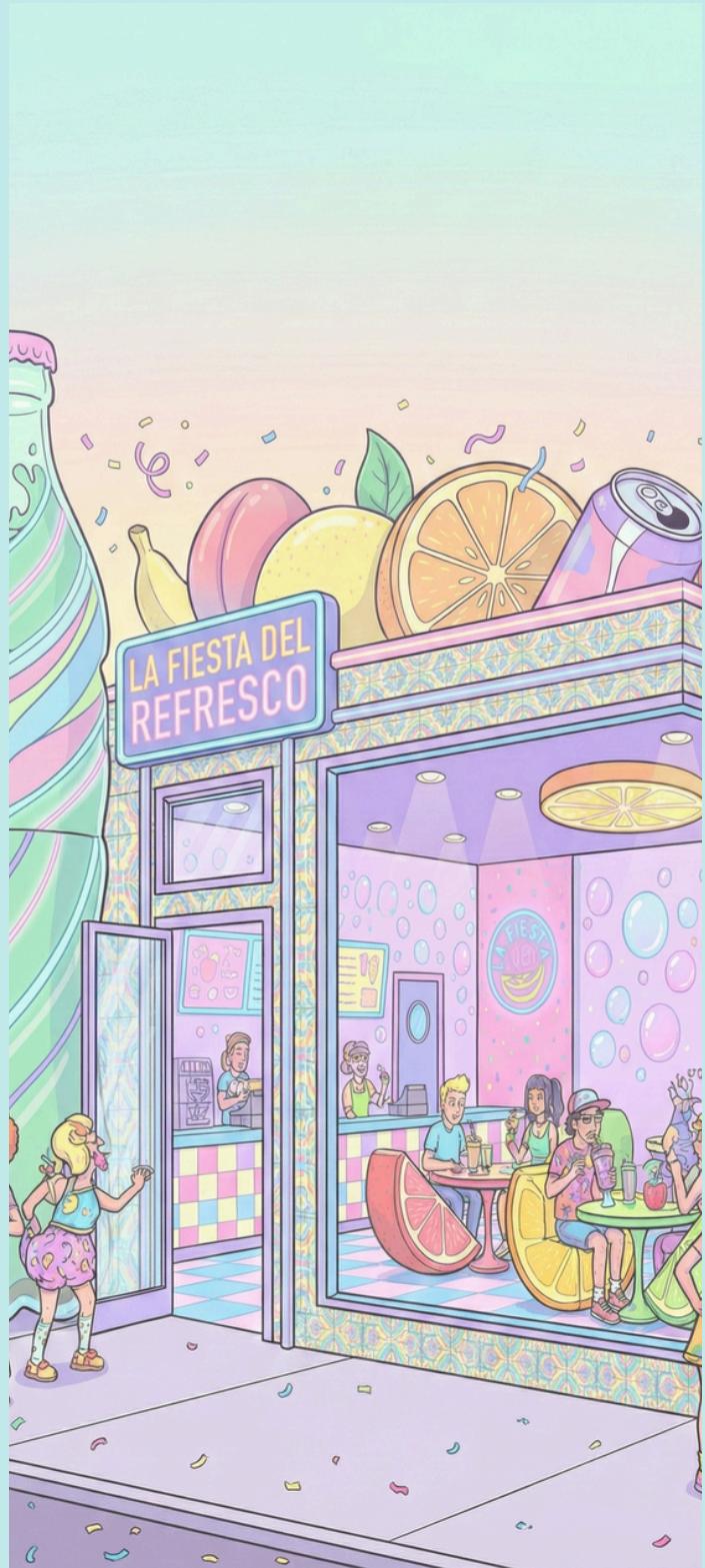


PROGRAMACION  
ESTRUCTURADA

BRAVO DUARTE LEONARDO ANTONIO

DE LA CUEVA GUTIERREZ ANGEL ALBERTO

# REFRESCUERIA



---

MANUAL  
TECNICO

# ÍNDICE

- 01** NUESTRO PROYECTO
- 02** REQUERIMIENTOS
- 03** VARIABLES
- 06** FUNCIONES
- 06** EXPLICACION
- 07** NUESTRO EQUIPO



# NUESTRO PROYECTO



---

ESTE PROYECTO FINAL TIENE LA FINALIDAD DE PONER EN PRACTICA LOS CONOCIMIENTOS ADQUIRIDOS DURANTE EL CURSO DE PROGRAMACION ESTRUCTURADA, CON ESTO MISMO, SATISFACER LA RUBRICA DE CALIFICACION DE ESTE PROYECTO.

ESTE MANUAL  
ESTÁ  
PENSADO  
PARA SERVIR  
COMO UNA  
GUÍA  
PRÁCTICA QUE  
AYUDE A  
COMPRENDER  
MEJOR CADA  
PARTE DEL  
PROCESO Y  
FACILITE EL  
USO DE LA  
INFORMACIÓN  
PASO A PASO.

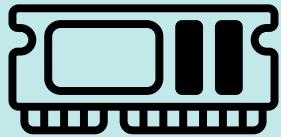
# REQUERIMIENTOS DEL SISTEMA

LO ESENCIAL PARA EJECUTAR EL PROGRAMA

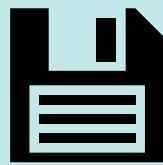
---

Este programa pide como requisitos mínimos:

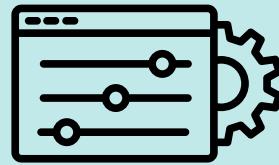
- REQUISITO: ESPECIFICACIÓN MÍNIMA
- MEMORIA RAM: 128 MB
- ESPACIO EN DISCO: 50 MB (PARA ALMACENAMIENTO DE REGISTROS)
- PROCESADOR: INTEL PENTIUM 4 / AMD ATHLON O SUPERIOR
- SISTEMA OPERATIVO: WINDOWS 7 / 10 / 11
- ARQUITECTURA: X86 (32 BITS) O X64 (64 BITS)



RAM: 128MB



MEMORIA: 50MB



SO: WINDOWS

# VARIABLES

VARIABLES DENTRO DEL CODIGO Y  
SU UTILIDAD

---

## LISTA DE VARIABLES

- **Estructura clase**

- 
- Variable / Tipo / Descripción
- sabor / char[50] / Sabor del producto
- topping / char[50] / Topping (nieves)
- cono / char[50] / Cono (nieves)
- tamano / char[50] / Tamaño (aguas)
- precio / int / Precio del producto

- **Estructura factura**

- Variable | Tipo | Descripción
- clave | int | ID único de la factura
- clase | texto | Paleta, Nieve o Agua
- sabor | texto | Sabor
- top | texto | Topping si aplica
- cono2 | texto | Cono si aplica
- cant | int | Cantidad vendida
- pre | int | Precio unitario
- total | int | Total calculado
- ultimo | int | Control interno
- tam | texto | Tamaño (solo para aguas)

- **Estructura producto**

- Variable | Tipo | Descripción
- clase | char[50] | Categoría del producto
- key | classe | Datos esenciales del producto
- F | factura | Factura vinculada si corresponde

- Variables dentro del main
- 
- Variable | Tipo | Uso
- Registro | producto | Guarda datos del producto
- co | int | contador para número de productos
- opcion | int | Elección del menú (en el switch case)

- Variables en la función altas

- 
- Variable | Tipo
- binario | FILE\*
- opc | int
- opc2 | int
- opc3 | int
- opc4 | char de largo medio
- selection | int

- Variables de la función ordenar

- 
- Variable | Tipo
- binario | FILE\*
- e, e2, aux | producto
- x, y, n | int

- Variables de la función consultas

- 
- Variable | Tipo
- binario | FILE\*
- e | producto

- Variables de la función consultas por clasificación

- 
- Variable | Tipo
- binario | FILE\*
- e | producto
- opc | int

- Variables de la función venta y generación de facturas (ventafac)

- Variable / Tipo
- binario / FILE\*
- fac / FILE\*
- P / producto
- F / factura
- ultimo / factura
- opcion / int
- bandera / int
- a / int
- cant / int
- peso / long

- Variables de la función consulta de facturas (consfact)
  - 
  - Variable | Tipo
  - binario | FILE\*
  - fac | FILE\*
  - F | factura

- Variables de la función cancelar facturas (cancelar)

- Variable | Tipo
- fac | FILE\*
- temporal | FILE\*
- F | factura
- pos | int
- facturab | int

# LISTA DE FUNCIONES

FUNCIONES DENTRO DEL CODIGO Y  
SU UTILIDAD

- Función | Retorno | Descripción
- main() | int | Muestra el menú y llama funciones principales
- altas(...) | int | Da de alta productos y los guarda
- ordenar(...) | void | Ordena por sabor y clase
- consultas(...) | void | Consulta general
- conclas(...) | void | Consulta por clasificación
- ventafac(...) | int | Genera factura
- consfact(...) | void | Muestra facturas
- anula(...) | int | Elimina una factura

# EXPLICACION DEL CODIGO LINEA POR LINEA

- Librerías:

- 01 #include <stdio.h> : Incluye las funciones de entrada y salida estándar: printf(), scanf(), fopen(), etc.
- 02 #include <string.h>: Permite usar funciones para cadenas como strcpy(), strcmp(), gets(), etc.
- 03 #include <ctype.h>: Incluye funciones para manipular caracteres (mayúsculas, minúsculas).
- 04 #include <locale.h>: Permite modificar la configuración regional (acentos, idioma).

```
5     // Borrador de estructuras
6
7     typedef struct {
8         char sabor[50];
9         char topping[50];
10        char cono[50];
11        char tamano[50];
12        int precio;
13    } classe;
14    typedef struct {
15        int clave;
16        char clase[50];
17        char sabor[50];
18        char top[50];
19        char cono2[50];
20        int cant;
21        int pre;
22        int total;
23        int ultimo;
24        char tam[50];
25    } factura;
26    typedef struct {
27        char clase[50];
28        classe key;
29        factura F;
30    } producto;
```

- 05 Comentario que indica el inicio de las funciones.
- 07 Esta es la primera es la primer estructura llamada "classe", aquí se pone información de los productos (dependiendo si es agua, paleta o nieve), y tiene variables de sabor, topping, cono, tamano y precio, porque se reutilizan con algunos requisitos.
- 13
- 14 Esta es la segunda estructura llamada "factura", aquí se pone información de los productos vendidos para después generar una factura (dependiendo si es agua, paleta o nieve), y tiene variables de sabor, topping, cono, tamano y precio, porque se reutilizan con algunos requisitos.
- 25
- 26 Esta es la tercer estructura llamada "producto", aquí se pone las estructuras ya mencionadas para hacer una anidación, nomas aclarando que la linea 27 es "clase", para que escriba que producto es y hacer una comparacion a futuro.
- 30

```
32 // Prototipos de funciones  
33 int altas(producto Registro, int co);  
34 void consultas(producto Registro, int co);  
35 void conclas(producto Registro, int co);  
36 void ordenar(producto Registro, int co);  
37 int ventafac(producto Registro, int co);  
38 void consfact(producto Registro, int co);  
39 int cancelar(producto Registro, int co);
```

32

Aquí se declaran las funciones para poder utilizarlas en el main (se ponen primero que el main), ya después las llamas en el main y mas adelante se hacen.

39

- 40 //Menu principal: es un comentario para guiarnos y saber el inicio.
- 41 int main (){ liniio del programa.
- 42 system ("color 9f"); aqui es para cambiar el color de la consola.
- 43 producto Registro; , Crea una variable tipo producto para capturas.
- 44 int co = 0; contador para que sume el total de registros.
- 45 int opcion; variable para seleccionar en el menú mas adelante.
- 46 do { Inicia un ciclo que repetirá el menú hasta que el usuario elija 0.
- 47 printf("\n-----Menu-----\n"); es para hacer una separacion, el encabezado del menu
- 48 printf("1 || Altas\n"); es la primera opción, y sirve para registrar una venta (producto).
- 49 printf("2 || Consulta ordenada\n"); es para ver los productos que ya registraste, pero de manera ordenada.
- 50 printf("3 || Consulta por clasificacion\n"); consulta el producto por tipo.
- 51 printf("4 || Venta y generacion de factura\n"); miras las ventas para que genera una factura.

```

52     printf("5  || Consulta de facturas\n");
53     printf("6  || Cancelar factura\n");
54     printf("0  || Salir\n");
55     puts("-----");
56     printf("Ingrese una opcion: ");
57     scanf("%d", &opcion);
58     switch (opcion) {
59         case 1:
60             co = altas(Registro, co);
61             break;
62         case 2:
63             ordenar(Registro, co); consultas(Registro, co);
64             break;
65         case 3:
66             ordenar(Registro, co); conclas(Registro, co);
67             break;
68         case 4:
69             ventafac(Registro, co);
70             break;
71         case 5:
72             consfact(Registro, co);
73             break;
74         case 6:
75             cancelar(Registro, co);
76             break;
77         default:
78             printf("Opcion invalida\n");
79     }

```

52. Función de consulta de facturas.
53. Función para cancelar una factura.
54. Opción para salir.
55. Pie de pagina, separación del menú.
56. Pide la opcion al usuario.
57. Lee la opción ingresada.
58. Empieza el switch case para poner las funciones y pueda escoger.
59. Caso 1
60. Para ejecuta altas(), actualiza co.
61. Finaliza el caso con el break;
62. Empieza el caso 2.
63. Para ejecutar Ordena y consulta.
64. Finaliza el caso 2.
65. Inicia el caso 3.
66. Ejecuta Consulta por clasificación.
67. Finaliza el caso 3.
68. Empieza el caso 4.
69. Para ejecutar Genera factura.
70. Rompe el caso 4.
71. Empieza el caso 5.
72. Ejecuta la consulta de facturas.
73. Finaliza el caso 5.
74. Inicia el caso 6.
75. Para ejecutar la cancelación de una factura.
76. Rompe el caso 6.
77. y 78. Para indicar que termino o hubo un error al escoger.

```

80     } while (opcion != 0);
81     return 0;
82 }
83
84 // Funciones
85 int altas(producto Registro, int co) {
86     FILE *binario;
87     int opc, opc2, opc3;
88     char opc4[50];
89     int selection;
90     binario = fopen("datos.dat", "ab");
91     //POSIBLE OPTIMIZACION
92     do {
93         //Dar de alta y se lleva a seleccion de sabor
94         puts("Ingrese que tipo de producto quiere agregar: ");
95         puts("|| 1: Paleta || 2: Nieve || 3: Agua ||");
96         scanf("%d", &selection);
97         fflush(stdin);
98         //Aqui se copia el tipo de producto a la estructura
99         if (selection == 1) {
100             strcpy(opc4, "Paleta");
101         }
102         else if (selection == 2) {
103             strcpy(opc4, "Nieve");
104         }
105         else if (selection == 3) {
106             strcpy(opc4, "Agua");
107         }

```

80.Para finalizar el ciclo de do{}while hasta que la opción sea 0.

81. Fin del programa.

82. hasta el 83 nada importante.

84.Comentario de (desarrollo de las funciones).

85.Declaración de la función: Registro: estructura donde se guarda temporalmente el producto.

co: cantidad de registros existentes (contador).

86. Abres el archivo donde se va a guardar los productos.

87. Declaración de variables, que se usan en menús internos.

88. Variable auxiliar, para copiar el nombre de la clase.

89. selection: determina si es Paleta / Nieve / Agua.

90. Abre o crea archivo binario para agregar (append binary). Cada registro nuevo se escribe al final del archivo.

91. Comentario de ayuda para nosotros.

92. Permite dar de alta varios productos sin salir al menú principal.

93. Comentario guia.

94. puts("Ingrese que tipo de producto quiere agregar: "); Muestra mensaje solicitando tipo de producto.

95.puts("|| 1: Paleta || 2: Nieve || 3: Agua ||"); Lista las opciones disponibles.

96.scnf("%d", &selection); Captura la opción seleccionada.

97.fflush(stdin); Limpia el buffer de entrada para evitar errores en la lectura de cadenas.

98. Comentario guia de donde se copia el producto

99.if (selection == 1) { strcpy(opc4, "Paleta"); }. Copia el tipo Paleta a la variable opc4.

100. Copia el tipo Paleta a la variable opc4.

101. Llave.

102, 103 y 104: Copia el tipo Nieve.

105.al 107. Copia el tipo Agua.

```

108     strcpy(Registro.clase, opc4);
109     //En caso de ser tal cosa, hacer:
110     switch (selection) {
111         case 1:
112             puts("Ingrese el sabor de la paleta: ");
113             puts("|| 1 para ver lista de sabores || 2 para ingresar manualmente ||");
114             scanf("%d", &opc2);
115             switch (opc2) {
116                 case 2:
117                     while(getchar() != '\n');
118                     puts("Ingresa el sabor: ");
119                     gets(Registro.key.sabor);
120                     break;
121                 case 1:
122                     do {
123                         puts("|| 1: Fresa || 2: Chocolate || 3: Vainilla ||"); //Agregar mas
124                         scanf("%d", &opc3);
125                         switch (opc3) {
126                             case 1:
127                                 strcpy(Registro.key.sabor, "Fresa");
128                                 break;
129                             case 2:
130                                 strcpy(Registro.key.sabor, "Chocolate");
131                                 break;
132                             case 3:
133                                 strcpy(Registro.key.sabor, "Vainilla");
134                                 break;

```

108. Guarda el tipo de producto dentro de la estructura principal.

109. Comentario guía.

110. Dependiendo del tipo de producto ejecuta un bloque distinto.

111. Inicia configuración para Paleta.

112. Pide un sabor.

113. Permite ver lista o ingresar sabor manualmente.

114. Lee la elección de sabor.

115. Inicia selección del modo de ingreso.

116. Opción para escribir sabor manualmente.

117. Limpia el buffer antes de leer texto.

118. Solicita el nombre del sabor.

119. Guarda el nombre ingresado en la estructura.

120. Termina el caso.

121. Si se eligen sabores de lista predeterminada.

122. Inicia un ciclo para mostrar la lista.

123. Muestra lista y captura opción.

124. Lee la elección.

125. Dependiendo de la opción, copia el sabor.

126. Inicia el caso 1.

127. Copia el sabor

128. Termina el caso 1.

129. Empieza el caso 2.

130. Copia el sabor

131. Se rompe el caso 2.

132. Empieza el caso 3.

133. Se copia el sabor.

134. Se rompe el caso 3.

```

135                     default:
136                         puts("Opcion invalida");
137                         break;
138                     }
139                 } while (opc3 != 1 && opc3 != 2 && opc3 != 3);
140                 break;
141             }
142             fflush(stdin);
143             puts("Ingrese el precio que se le dara: ");
144             scanf("%d", &Registro.key.precio);
145             break;
146         case 2:
147             while(getchar() != '\n');
148             puts("Ingrese el sabor de la nieve: ");
149             puts("|| 1 para ver lista de sabores || 2 para ingresar manualmente ||");
150             scanf("%d", &opc2);
151             switch (opc2) {
152                 case 2:
153                     while(getchar() != '\n');
154                     puts("Ingresa el sabor: ");
155                     gets(Registro.key.sabor);
156                     break;
157                 case 1:
158                     do {
159                         puts("|| 1: Menta || 2: Chocolate || 3: Vainilla ||"); //Agregar mas
160                         scanf("%d", &opc3);
161                         switch (opc3) {

```

135 y 136. Muestra un mensaje de error.

137. Rompe el default.

138. Llave.

139. Valida que la opción sea correcta.

140. Rompe el ciclo o lo termina.

141. Lave.

142. Limpia el buffer.

143. Solicita el precio del producto.

144. Guarda el precio en la estructura.

145. Fin del caso Paleta.

146. Inicia el caso de la nieve.

147. Limpia el buffer.

148. Pide un sabor.

149. Permite ver lista o ingresar sabor manualmente.

150. Lee la elección de sabor.

151. Inicia selección del modo de ingreso.

152. Opción para escribir sabor manualmente.

153. Limpia el buffer antes de leer texto.

154. Solicita el nombre del sabor.

155. Guarda el nombre ingresado en la estructura.

156. Termina el caso.

157. Empieza caso 1.

158. Si se eligen sabores de lista predeterminada.

159. Inicia un ciclo para mostrar la lista.

160. Muestra lista y captura opción.

```

161         switch (opc3) {
162             case 1:
163                 strcpy(Registro.key.sabor, "Menta");
164                 break;
165             case 2:
166                 strcpy(Registro.key.sabor, "Chocolate");
167                 break;
168             case 3:
169                 strcpy(Registro.key.sabor, "Vainilla");
170                 break;
171             default:
172                 puts("Opcion invalida");
173                 break;
174             }
175         } while (opc3 != 1 && opc3 != 2 && opc3 != 3);
176         break;
177     }
178     puts("Ingrese el topping deseado: ");
179     puts("|| 1 para ver lista de toppings || 2 para ingresar manualmente ||");
180     scanf("%d", &opc2);
181     switch (opc2) {
182         case 2:
183             while(getchar() != '\n');
184             puts("Ingresa el topping: ");
185             gets(Registro.key.topping);
186             break;
187         }

```

161. Inicia selección del sabor para Nieve.
162. Inicia el caso 1.
163. Si el usuario elige 1, asigna el sabor Menta.
164. Se rompe el caso 1.
165. Empieza el caso 2.
166. Si elige 2, asigna Chocolate.
167. Rompe el caso 2.
168. Inicia el caso 3.
169. Si elige 3, asigna Vainilla.
170. Rompe el caso 3.
- 171 y 172. Mensaje de error si no elige una opción válida.
173. Rompe el bloque
174. Llave.
175. Repite el menú hasta elegir una opción válida.
176. Rompe el ciclo
177. Llave.
178. Solicita el topping para la nieve.
179. Muestra las opciones.
180. Captura la opcion.
181. Inicia el menú de topping.
182. Inicia el caso 2.
183. Limpia el buffer.
184. Solicita el nombre del topping.
185. Guarda el topping ingresado.
186. Rompe el caso 2.

```
187         case 1:  
188             do {  
189                 puts("|| 1: Chispas de chocolate || 2: Nuez || 3: Gomitas ||"); //Agregar mas  
190                 scanf("%d", &opc3);  
191                 switch (opc3) {  
192                     case 1:  
193                         strcpy(Registro.key.topping, "Chispas de chocolate");  
194                         break;  
195                     case 2:  
196                         strcpy(Registro.key.topping, "Nuez");  
197                         break;  
198                     case 3:  
199                         strcpy(Registro.key.topping, "Gomitas");  
200                         break;  
201                     default:  
202                         puts("Opcion invalida");  
203                         break;  
204                 }  
205             } while (opc3 != 1 && opc3 != 2 && opc3 != 3);  
206             break;  
207         }  
208     }
```

187. Si quiere elegir de lista.

188. Empieza el ciclo

189. Muestra la lista de toppings y captura opción.

190. Captura la opción del usuario.

191. Inicia selección de topping.

192. Empieza el caso 1.

193. Asigna "Chispas de chocolate".

194. Se rompe el caso 1.

195. Empieza el caso 2.

196. Asigna "Nuez".

197. Se rompe el caso 2.

198. Inicia el caso 3.

199. Asigna "Gomitas".

200. Se rompe el caso 3.

201 hasta el 203, es un mensaje de error, la opcion es invalida y cierra el bloque.

204. Llave.

205. Valida la selección.

206. Rompe el bloque.

207. Llave.

```

208     fflush(stdin);
209     puts("Ingrese el tipo de cono deseado: ");
210     puts("|| 1 para ver lista de conos || 2 para ingresar manualmente ||");
211     fflush(stdin);
212     scanf("%d", &opc2);
213     switch (opc2) {
214         case 2:
215             while(getchar() != '\n');
216             puts("Ingresa el cono: ");
217             gets(Registro.key.cono);
218             break;
219         case 1:
220             do {
221                 puts("|| 1: Normal || 2: Waffle || 3: Chocolate ||"); //Agregar mas
222                 scanf("%d", &opc3);
223                 switch (opc3) {
224                     case 1:
225                         strcpy(Registro.key.cono, "Normal");
226                         break;
227                     case 2:
228                         strcpy(Registro.key.cono, "Waffle");
229                         break;
230                     case 3:
231                         strcpy(Registro.key.cono, "Chocolate");
232                         break;
233                     default:
234                         puts("Opcion invalida");
235                         break;

```

208. Limpia buffer.
209. Solicita el tipo de cono.
210. Muestra opciones.
211. Limpia buffer nuevamente.
212. Captura la elección.
213. Inicia menú de conos.
214. Inicia el caso 2.
215. Limpia el buffer.
216. Pide el tipo de cono.
217. Guarda la entrada.
218. Rompe el caso 2.
219. Se crea el caso 1.
220. Inicia el ciclo.
221. Muestra la lista de conos.
222. Captura la selección.
223. Inicia selección.
224. Inicia el caso 1.
225. Cono Normal.
226. Se rompe el caso 1.
227. Inicia el caso 2.
228. Cono de waffle.
229. Se rompe el caso 2.
230. Inicia el caso 3.
231. Cono de chocolate.
232. Se rompe el caso 3.
- 233 a 235 es para mostrar un mensaje de error y la linea 235 rompe el default.

```

236
237         }
238     } while (opc3 != 1 && opc3 != 2 && opc3 != 3);
239     break;
240 }
241 fflush(stdin);
242 puts("Ingrese el precio que se le dara: ");
243 scanf("%d", &Registro.key.precio);
244 break;
245 case 3:
246     while(getchar() != '\n');
247     puts("Ingrese el sabor de la agua: ");
248     puts("|| 1 para ver lista de sabores || 2 para ingresar manualmente ||");
249     fflush(stdin);
250     scanf("%d", &opc2);
251     switch (opc2) {
252         case 2:
253             while(getchar() != '\n');
254             puts("Ingresa el sabor: ");
255             gets(Registro.key.sabor);
256             break;
257         case 1:
258             do {
259                 puts("|| 1: Limon || 2: Frutas || 3: Agua Natural ||"); //Agregar mas
260                 scanf("%d", &opc3);

```

236. Llave.

237. Repite hasta elegir bien.

238. Rompe todo el break.

239. Llave.

240. Limpia buffer.

241. Pide el precio.

242. Almacena el precio.

243. Termina el caso 2 por completo.

244. Inicia el caso 3.

245. Limpia buffer para texto.

246. Solicita el sabor.

247. Muestra opciones.

248. Limpia el buffer.

249. Captura la opción.

250. Inicia menú de sabor.

251. Inicia el caso 2.

252. Limpia el buffer.

253. Entrada manual, solicita sabor.

254. Lo guarda.

255. Rompe el caso 2.

256. Inicia el caso 1.

257. Inicia el ciclo.

258. Muestra lista.

259. Captura.

```

260         switch (opc3) {
261             case 1:
262                 strcpy(Registro.key.sabor, "Limon");
263                 break;
264             case 2:
265                 strcpy(Registro.key.sabor, "Frutas");
266                 break;
267             case 3:
268                 strcpy(Registro.key.sabor, "Agua Natural");
269                 break;
270             default:
271                 puts("Opcion invalida");
272                 break;
273             }
274         } while (opc3 != 1 && opc3 != 2 && opc3 != 3);
275         break;
276     }
277     fflush(stdin);
278     printf("Ingrese el tamano deseado: ");
279     puts("|| 1 para ver lista de tamanos || 2 para ingresar manualmente ||");
280     scanf("%d", &opc2);
281     switch (opc2) {
282         case 2:
283             while(getchar() != '\n');
284             printf("Ingresa el tamano: ");
285             gets(Registro.key.tamano);
286             break;

```

260. Selección de sabor.
261. Inicia el caso 1.
262. Asigna Limón.
263. Rompe el caso 1.
264. Inicia el caso 2.
265. Asigna Frutas.
266. Rompe el caso 2.
267. Inicia el caso 3.
268. Asigna agua natural.
269. Rompe el caso 3.
270. al 273 es un default para mostrar un mensaje de error, que no es una opción valida y rompe el default.
274. Se cierra el ciclo hasta que elija bien.
275. Rompe todo.
276. Llave.
277. Limpia buffer.
278. Solicita tamaño.
279. Muestra opciones.
280. Captura selección.
281. Inicia un nuevo switch para el tamaño del agua
282. Inicia el caso 2.
283. Limpia el buffer.
284. Solicitud el tamaño.
285. Captura el tamaño.
286. Rompe el break.

```

287         case 1:
288             do {
289                 puts("|| 1: 250ml || 2: 500ml || 3: 1L ||"); //Agregar mas
290                 scanf("%d", &opc3);
291                 switch (opc3) {
292                     case 1:
293                         strcpy(Registro.key.tamano, "250ml");
294                         break;
295                     case 2:
296                         strcpy(Registro.key.tamano, "500ml");
297                         break;
298                     case 3:
299                         strcpy(Registro.key.tamano, "1L");
300                         break;
301                     default:
302                         puts("Opcion invalida");
303                         break;
304                 }
305             } while (opc3 != 1 && opc3 != 2 && opc3 != 3);
306             break;
307         }
308         fflush(stdin);
309         puts("Ingrese el precio que se le dara: ");
310         scanf("%d", &Registro.key.precio);
311         break;
312     }
313     fwrite(&Registro, sizeof(producto), 1, binario);
314     co++;
315     printf("\nGuardado exitoso\n");

```

287. Inicia el caso 1.

288. Inicia el ciclo para el tamaño del agua.

289. Muestra lista.

290. Captura la selección.

291. Inicia un switch

292. Inicia el caso 1.

293. Asigna 250ml.

294. Se rompe el caso 1.

295. Inicia el caso 2.

296. Asigna 500ml.

297. Se rompe el caso 2.

298. Inicia el caso 3.

299. Asigna 1L.

300. Se rompe el caso 3.

301 al 304 es un default para mostrar un mensaje de error, que no es una opción valida y rompe el default

305. se cierra el ciclo hasta que elija bien.

306. Se rompe el switch.

307. Llave.

308. Limpia el buffer.

309. Solicita el precio que se le dará.

310. Captura lo que se pide.

311. Se rompe todo.

312.

313. Llave.

314. Guarda el registro en el archivo.

315. Incrementa el contador.

316. Mensaje de confirmación.

317. Pregunta para seguir.

318. Opciones.

319. Captura la opción.

320. Repite si el usuario quiere agregar más productos.

321. Cierra el archivo binario.

322. Devuelve el conteo de registros.

323. Llave final de la función.

```
325     void ordenar(producto Registro, int co) {
326         FILE *binario;
327         producto e, e2, aux;
328         binario = fopen("datos.dat", "rb+");
329         int x,y,n;
330
331         if (binario == NULL) {
332             printf("Error al abrir el archivo\n");
333             return;
334         }
335         else {
336             //Esto calcula cuantos registros hay
337             fseek(binario, 0, SEEK_END);
338             n = ftell(binario) / sizeof(producto);
339             rewind(binario);
340
341             //Ordenacion burbuja
342             for (x = 0; x<n; x++) {
343                 for (y = 0; y<n; y++) {
344                     //
345                     fseek(binario, y*sizeof(producto), 0);
346                     fread(&e, sizeof(producto), 1, binario);
347                     fseek(binario, (y+1)*sizeof(producto), 0);
348                     fread(&e2, sizeof(producto), 1, binario);
349                     //Ordena por nombre de sabor
350                     if (strcmp(e.key.sabor, e2.key.sabor) > 0) {
351                         aux = e;
352                         e = e2;
353                         e2 = aux;
```

325. Inicio de la función ordenar. Recibe un registro y el contador, aunque aquí no se usan directamente.

326. Declaración del apuntador al archivo que contiene los productos.

327. Tres estructuras auxiliares:

1. e → producto actual

2. e2 → producto siguiente

3. aux → variable temporal para intercambiar datos

328. Abre el archivo en modo lectura–escritura binaria. Permite leer y volver a escribir los registros ordenados.

329. Variables para ciclos (x,y) y para la cantidad total de registros (n).

330.

331. Verifica si el archivo no abrió correctamente.

332. Mensaje de error.

333. Finaliza la función.

334. Fin del if.

335. Inicia el bloque cuando el archivo sí abrió correctamente.

336. Comentario guia.

337. Mueve el puntero al FINAL del archivo.

338. Divide los bytes totales entre el tamaño de un producto para saber cuántos registros hay.

339. Regresa el puntero al inicio del archivo.

- 340. Nada
- 341. Comentario de inicio de la burbuja
- 342. Primera parte del ciclo burbuja.
- 343. Ciclo interno, recorre cada par de elementos.
- 344.
- 345. Posiciona el cursor en el registro y.
- 346. Lee el registro y lo guarda en e.
- 347. Se mueve al siguiente registro.
- 348. Lee el registro siguiente y lo guarda en e2.
- 349. Comentario guia.
- 350. Compara los sabores.
- 351. Guarda e temporalmente.
- 352. Coloca e2 en la posición de e.
- 353. Termina el intercambio.

```

350         if (strcmp(e.key.sabor, e2.key.sabor) > 0) {
351             aux = e;
352             e = e2;
353             e2 = aux;
354
355             fseek(binario, y*sizeof(producto), 0);
356             fwrite(&e, sizeof(producto), 1, binario);
357             fseek(binario, (y+1)*sizeof(producto), 0);
358             fwrite(&e2, sizeof(producto), 1, binario);
359         }
360         //Ordena por tipo
361         if (strcmp(e.clase, e2.clase) > 0) {
362             aux = e;
363             e = e2;
364             e2 = aux;
365
366             fseek(binario, y*sizeof(producto), 0);
367             fwrite(&e, sizeof(producto), 1, binario);
368             fseek(binario, (y+1)*sizeof(producto), 0);
369             fwrite(&e2, sizeof(producto), 1, binario);
370         }
371     }
372 }
373 fclose(binario);
374 }
375 }
```

355. Posiciona al inicio del registro actual.

356. Escribe el nuevo registro ordenado.

357. Se mueve al siguiente registro.

358. Guarda el segundo registro también ordenado.

359. Llave.

360. Comentario ordenación por tipo.

361. Compara la clase (Paleta, Nieve, Agua). Si están desordenados, intercambia igual que antes.

362. Guarda temporalmente.

363. Intercambia.

364. Finaliza intercambio.

365.

366. Apunta al registro actual.

367. Reescribe e ordenado.

368. Apunta al siguiente.

369. Escribe e2 ordenado.

370. Llave.

371. Llave.

372. Llave.

373. Cierras archivo

374. Llave.

375. Llave.

```

void consultas(producto Registro, int co) {
    FILE *binario; producto e;
    binario = (fopen("datos.dat", "rb"));
    if (binario == NULL) {
        printf("Error al abrir el archivo\n");
        return;
    }
    else {
        puts("-----Menu-----");
        printf("*****\n");
        while (fread(&e, sizeof(producto), 1, binario)>0) {
            printf("|| Tipo: %s\n", e.clase);
            if (strcmp(e.clase, "Paleta") == 0) {
                printf("|| Sabor: %s\n", e.key.sabor);
                printf("|| Precio: %d\n", e.key.precio);
                printf("*****\n");
            }
            else if (strcmp(e.clase, "Nieve") == 0) {
                printf("|| Sabor: %s\n", e.key.sabor);
                printf("|| Topping: %s\n", e.key.topping);
                printf("|| Cono: %s\n", e.key.cono);
                printf("|| Precio: %d\n", e.key.precio);
                printf("*****\n");
            }
            else if (strcmp(e.clase, "Agua") == 0) {
                printf("|| Sabor: %s\n", e.key.sabor);
                printf("|| Tamano: %s\n", e.key.tamano);
                printf("|| Precio: %d\n", e.key.precio);
                printf("*****\n");
            }
        }
        puts("-----");
        fclose(binario);
    }
}

```

## nicio de la función

- 1.void consultas() {
2. Inicio de la función que muestra el menú de consultas para el usuario.
- 3.int opcion;
4. Variable para almacenar la opción que el usuario elija.
- 5.do {

6. Inicio de un ciclo do-while para mostrar el menú repetidamente hasta que el usuario decida salir.

## Menú de opciones

- 1.printf("\nMENU CONSULTAS\n");
2. Imprime el título del menú.
- 3.printf("1. Consulta por clase\n");
4. Opción para consultar productos por su clase (Paleta, Nieve, Agua).
- 5.printf("2. Consulta por sabor\n");
6. Opción para consultar productos por sabor.
- 7.printf("3. Consulta por precio\n");
8. Opción para consultar productos según su precio.
- 9.printf("O. Salir\n");
10. Opción para salir del menú de consultas.
- 11.printf("Ingrese su opcion: ");
12. Solicita que el usuario ingrese su opción.
- 13.scnf("%d", &opcion);
14. Lee la opción ingresada por el usuario y la guarda en la variable opcion.
15. Evaluación de la opción elegida
- 16.exit(0); }

1.switch(opcion) {

2. Estructura switch para evaluar la opción ingresada.

Caso 1: Consulta por clase

1.case 1:

2. Si el usuario eligió opción 1.

3.conclas();

4. Llama a la función conclas() que muestra productos filtrados por clase.

5.break;

6. Termina este caso y sale del switch.

Caso 2: Consulta por sabor

1.case 2:

2. Opción 2 elegida.

3.consabor();

4. Llama a la función consabor() para consultar productos por sabor.

5.break;

6. Sale del switch después de esta acción.

Caso 3: Consulta por precio

1.case 3:

2. Opción 3 seleccionada.

3.conprecio();

4. Llama a la función conprecio() para buscar productos según precio.

5.break;

6. Finaliza este caso.

Caso 0: Salir

1.case 0:

2. Opción 0 para salir.

3.printf("Saliendo del menu consultas...\n");

4. Muestra mensaje de salida.

5.break;

6. Termina el ciclo.

Caso por defecto (opción inválida)

1.default:

2. Si la opción no es ninguna de las anteriores.

3.printf("Opcion no valida, intente de nuevo.\n");

4. Muestra mensaje de error.

5.break;

6. Regresa para que el usuario ingrese una opción válida.

Fin del ciclo do-while

1.} while(opcion != 0);

2. Repite el menú mientras la opción sea diferente de 0 (salir).

Fin de la función

1.}

2. Cierre de la función consultas().

```

413 void conclas(producto Registro, int co) {
414     FILE *binario; producto e;
415     int opc;
416     binario = (fopen("datos.dat", "rb"));
417     if (binario == NULL) {
418         puts("Error al abrir el archivo");
419         return;
420     }
421     else {
422         printf("Ingresa la clase que quieras consultar: \n");
423         puts("|| 1: Paletas || 2: Nieve || 3: Agua ||");
424         //Esto solo sirve para el switch
425         scanf("%d", &opc);
426         puts("-----Menu-----");
427         switch (opc) {
428             case 1:
429                 puts("*****");
430                 while (fread(&e, sizeof(producto), 1, binario)>0) {
431                     if (strcmp(e.clase, "Paleta") == 0) {
432                         printf("|| Tipo: %s\n", e.clase);
433                         printf("|| Sabor: %s\n", e.key.sabor);
434                         printf("|| Precio: %d\n", e.key.precio);
435                         puts("*****");
436                     }
437                 }
438                 break;
439             case 2:
440                 puts("*****");
441                 while (fread(&e, sizeof(producto), 1, binario)>0) {
442                     if (strcmp(e.clase, "Nieve") == 0) {
443                         printf("|| Tipo: %s\n", e.clase);
444                         printf("|| Sabor: %s\n", e.key.sabor);
445                         printf("|| Topping: %s\n", e.key.topping);
446                         printf("|| Cono: %s\n", e.key.cono);
447                         printf("|| Precio: %d\n", e.key.precio);
448                         puts("*****");
449                 }

```

```

448         puts("*****");
449         while (fread(&e, sizeof(producto), 1, binario)>0) {
450             if (strcmp(e.clase, "Nieve") == 0) {
451                 printf("|| Tipo: %s\n", e.clase);
452                 printf("|| Sabor: %s\n", e.key.sabor);
453                 printf("|| Topping: %s\n", e.key.topping);
454                 printf("|| Cono: %s\n", e.key.cono);
455                 printf("|| Precio: %d\n", e.key.precio);
456                 puts("*****");
457             }
458             break;
459         case 3:
460             puts("*****");
461             while (fread(&e, sizeof(producto), 1, binario)>0) {
462                 if (strcmp(e.clase, "Agua") == 0) {
463                     printf("|| Tipo: %s\n", e.clase);
464                     printf("|| Sabor: %s\n", e.key.sabor);
465                     printf("|| Tamaño: %s\n", e.key.tamano);
466                     printf("|| Precio: %d\n", e.key.precio);
467                     puts("*****");
468                 }
469             }
470             break;
471         default:
472             puts("Opcion invalida");
473             break;
474         }
475     }
476 }

```

## Inicio de la función

- 1.void conclas(producto Registro, int co) {
2. Inicio de la función que permite consultar productos filtrados por su clasificación (clase).
- 3.FILE \*binario;
4. Declaración de un puntero para archivo binario, usado para abrir el archivo que contiene los productos.
- 5.producto e;
6. Variable temporal para almacenar cada producto leído desde el archivo.
- 7.int opc;
8. Variable para guardar la opción de clase que el usuario quiere consultar.

## Apertura del archivo

- 1.binario = (fopen("datos.dat", "rb"));
2. Se abre el archivo "datos.dat" en modo lectura binaria (rb). Aquí están almacenados los productos.
- 3.if (binario == NULL) {
4. Se verifica si hubo un error al abrir el archivo.
- 5.puts("Error al abrir el archivo");
6. Se imprime mensaje de error si no se pudo abrir.
- 7.return;
8. Se sale de la función ya que no se puede continuar sin el archivo.
- 9.}

## 10. Cierre del bloque if.

## Solicitud de la clase para consulta

- 1.else {
2. Si el archivo se abrió correctamente, se continúa.
- 3.printf("Ingresa la clase que quieras consultar: \n");
4. Se solicita al usuario ingresar qué clase de producto quiere consultar.
- 5.puts("|| 1: Paletas || 2: Nieve || 3: Agua ||");

1. Muestra las opciones posibles para la clase.

2. `scanf("%d", &opc);`

3. Se lee la opción elegida y se guarda en opc.

Impresión del menú de resultados

1. `puts("-----Menu-----");`

2. Muestra una línea separadora o encabezado para la salida.

Evaluación de la opción elegida (switch-case)

1. `switch (opc) {`

2. Se inicia la evaluación de la opción elegida para filtrar los productos por clase.

Caso 1: Paletas

1. `case 1:`

2. Si el usuario eligió la clase Paletas.

3. `puts("*****");`

4. Imprime separador visual para la salida.

5. `while (fread(&e, sizeof(producto), 1, binario) > 0) {`

6. Ciclo que lee cada registro del archivo hasta el final.

7. `if (strcmp(e.clase, "Paleta") == 0) {`

8. Compara si el campo clase del producto leído es "Paleta".

9. `printf("|| Tipo: %s\n", e.clase);`

10. Imprime el tipo de producto.

11. `printf("|| Sabor: %s\n", e.key.sabor);`

12. Imprime el sabor del producto.

13. `printf("|| Precio: %d\n", e.key.precio);`

14. Imprime el precio del producto.

15. `puts("*****");`

16. Imprime separador después de cada producto.

17. }

18. Cierre del if que verifica clase.

19. }

20. Cierre del while que recorre todos los productos.

21. `break;`

22. Termina este caso.

Caso 2: Nieve

1. `case 2:`

2. Para la clase Nieve.

3. `puts("*****");`

4. Separador.

5. `while (fread(&e, sizeof(producto), 1, binario) > 0) {`

6. Lee todos los productos.

7. `if (strcmp(e.clase, "Nieve") == 0) {`

8. Verifica que el producto sea Nieve.

9. `printf("|| Tipo: %s\n", e.clase);`

10. Imprime clase.

11. `printf("|| Sabor: %s\n", e.key.sabor);`

12. Imprime sabor.

13. `printf("|| Topping: %s\n", e.key.topping);`

14. Imprime topping.

15. `printf("|| Cono: %s\n", e.key.cono);`

16. Imprime tipo de cono.

1.printf("|| Precio: %d\n", e.key.precio);

2. Imprime precio.

3.puts("\*\*\*\*\*");

4. Separador.

5.}

6. Cierre del if.

7.}

8. Cierre del while.

9.break;

10. Termina este caso.

◆ Caso 3: Agua

1.case 3:

2. Para la clase Agua.

3.puts("\*\*\*\*\*");

4. Separador.

5.while (fread(&e, sizeof(producto), 1, binario) > 0) {

6. Lee todos los productos.

7.if (strcmp(e.clase, "Agua") == 0) {

8. Verifica que el producto sea Agua.

9.printf("|| Tipo: %s\n", e.clase);

10. Imprime clase.

11.printf("|| Sabor: %s\n", e.key.sabor);

12. Imprime sabor.

13.printf("|| Tamano: %s\n", e.key.tamano);

14. Imprime tamaño.

15.printf("|| Precio: %d\n", e.key.precio);

16. Imprime precio.

17.puts("\*\*\*\*\*");

18. Separador.

19.}

20. Cierre del if.

21.}

22. Cierre del while.

23.break;

24. Termina este caso.

◆ Caso por defecto: opción inválida

1.default:

2. Si la opción no es válida.

3.puts("Opcion invalida");

4. Muestra mensaje de error.

5.break;

6. Termina el switch.

◆ Fin del bloque else y función

1.}

2. Cierra el switch.

3.}

4. Cierra el bloque else de apertura de archivo.

5.}

6. Fin de la función conclas().

```

470 int ventafac(producto Registro, int co)
471 {
472     //apertura de archivos
473     FILE *binario;
474     FILE *fac;
475     producto P;
476     factura F;
477     factura ultimo;
478     //declaracion variables
479     int opcion=0, bandera=0, a=1;
480     int cant;
481
482     binario = fopen("datos.dat", "rb");
483     if (binario == NULL) {
484         puts("Error al abrir el archivo");
485         return 0;
486     }
487
488     fac = fopen("facturas.dat", "ab+");
489     if (fac == NULL) {
490         puts("Error al abrir el archivo");
491         return 0;
492     }
493
494     printf("\n-----Menu de productos disponibles-----\n");
495
496     //abre el archivo de altas para ver las opciones
497
498     while(fread(&P, sizeof(producto), 1, binario) >0){
499         if (strcasecmp(P.clase, "Paleta") == 0 || strcasecmp(P.clase, "Agua") == 0){
500             printf("|| %d. %s - %s - $%d\n", a, P.clase, P.key.sabor, P.key.precio);
501             a++;
502         }
503         else {
504             printf("|| %d. %s - %s - %s - $%d\n", a, P.clase, P.key.sabor, P.key.topping, P.key.cono, P.key.precio);
505             a++;
506         }
507     }
508
509     printf("\nSeleccione el numero del producto vendido: ");
510     scanf("%d", &opcion);
511
512     rewind(binario);
513     a=1;
514
515     //buscamos el producto elegido
516
517     while(fread(&P, sizeof(producto), 1, binario) >0){
518
519         if (a==opcion){
520             bandera =1;
521             break;
522         }
523         a++;
524     }
525
526     if(!bandera){
527         printf("\nNo se encontro el producto (ERROR)\n");
528         fclose(binario);
529         fclose(fac);
530         return 0;
531     }
532
533     printf("Ingrese la cantidad vendida: ");
534     scanf("%d", &cant);
535
536     //empezamos con la creacion de la factura
537
538     fseek(fac, 0, SEEK_END);
539     long peso = ftell(fac);
540
541     //validacion para ver si hay una factura creada o no
542     if(peso == 0){

```

```

543     P.F.clave=1;
544     fseek(fac, 0, SEEK_END);
545 }
546
547 //si ya hay una factura creada
548 else {
549     fseek(fac, -((long)sizeof(factura)), SEEK_END); // ir a la última factura (- para leer el ultimo digito)
550     fread(&ultimo, sizeof(factura), 1, fac); // lee la última factura
551     P.F.clave = ultimo.clave + 1; // siguiente ID
552 }
553
554 strcpy(P.F.clase, P.clase);
555 strcpy(P.F.sabor, P.key.sabor);
556 strcpy(P.F.top, P.key.topping);
557 strcpy(P.F.cono2, P.key.cono);
558 P.F.cant = cant;
559 P.F.pre = P.key.precio;
560 P.F.total = P.F.pre * cant;
561
562 //guarda la factura en el archivo
563 fwrite(&P.F, sizeof(factura), 1, fac);
564
565 //muestra la factura
566 printf("\n----- FACTURA GENERADA ----- \n");
567 printf("|| ID Factura: %d\n", P.F.clave);
568 printf("|| Producto: %s\n", P.F.clase);
569 printf("|| Sabor: %s\n", P.F.sabor);
570
571 if (strcmp(P.F.clase, "Nieve") == 0) {
572     printf("|| Topping: %s\n", P.F.top);
573     printf("|| Cono: %s\n", P.F.cono2);
574 }
575 if (strcmp(P.F.clase, "Agua") == 0) {
576     printf("|| Tamano: %s\n", P.key.tamano);
577 }
578
579 printf("|| Cantidad: %d\n", P.F.cant);
580 printf("|| Precio unitario: $%d\n", P.F.pre);

```

```
581     printf("|| TOTAL: %d\n", P.F.total);
582     printf("-----\n");
583
584     fclose(binario);
585     fclose(fac);
586     return 1;
587 }
588
589 void consfact(producto Registro, int co){
590     FILE *binario;
591     binario = (fopen("datos.dat", "rb"));
592     if (binario == NULL) {
593         printf("Error al abrir el archivo\n");
594         return;
595     }
596     FILE *fac;
597     factura F;
598     fac = (fopen("facturas.dat", "rb"));
599     if (fac == NULL) {
600         printf("Error al abrir el archivo\n");
601         return;
602     }
603     else {
604         puts("*****");
605
606         //lee el archivo de facturas
607         while (fread(&F, sizeof(factura), 1, fac)>0) {
608
609             //aqui se imprime el producto, o sea la factura
610             printf("|| ID de la factura: %d\n", F.clave);
611             printf("|| Producto: %s\n", F.clase);
612
613             if (strcmp(F.clase, "Paleta") == 0) {
614                 printf("|| Sabor: %s\n", F.sabor);
615                 printf("|| Precio: %d\n", F.precio);
616             }
617             else if (strcmp(F.clase, "Nieve") == 0) {
618                 printf("|| Sabor: %s\n", F.sabor);
619                 printf("|| Precio: %d\n", F.precio);
620             }
621         }
622     }
623 }
```

## Inicio de la función

- 1.int vantafac(producto Registro) {
  2. Inicio de la función. Esta función permite registrar una venta en factura y devuelve el número total de ventas registradas en la sesión.
  - 3.FILE \*binario;
  4. Puntero de archivo para abrir y manipular el archivo binario de ventas.
  - 5.int opc, opc2, opc3, co = 0;
  6. Variables enteras:
    - opc → opción general para repetir la venta o terminar.
    - opc2 → opción secundaria para elegir listas o ingreso manual.
    - opc3 → opción de submenú dentro de listas.
    - co → contador de ventas registradas.
- Inicio del ciclo de venta
- 1.do {
  2. Comienza un ciclo do-while para registrar ventas múltiples hasta que el usuario decida detenerse.
  - 3.binario = fopen("ventas.dat", "ab");
  4. Se abre el archivo ventas.dat en modo escritura binaria al final del archivo (ab) para agregar nuevas ventas.
  - 5.if(binario == NULL) {
  6. Se verifica si hubo un error al abrir el archivo.
  - 7.puts("Error al abrir el archivo");
  8. Muestra mensaje de error.
  - 9.return co;
  10. Si hay error, termina la función devolviendo la cantidad de ventas registradas hasta ese momento.
  - 11.}
  12. Cierre del bloque if.

## Solicitud del tipo de producto

- 1.puts("Seleccione el tipo de producto: ");
2. Solicita al usuario seleccionar el tipo de producto que se venderá.
- 3.puts("|| 1: Paleta || 2: Nieve || 3: Agua ||");
4. Muestra opciones disponibles.
- 5.scnf("%d", &opc);
6. Lee la opción seleccionada.

Switch para cada tipo de producto

1.switch(opc) {

2. Inicia el bloque switch según la clase del producto.

Caso 1: Paletas

1.case 1:

2. Si el usuario seleccionó Paletas.

3.while(getchar() != '\n');

4. Limpia el buffer de entrada para evitar que scanf y gets interfieran.

5.puts("Ingrese el sabor de la paleta: ");

6. Pregunta por el sabor.

7.puts("|| 1 para ver lista de sabores || 2 para ingresar manualmente ||");

8. Opción de usar lista predefinida o ingreso manual.

9.sccanf("%d", &opc2);

10. Lee la opción secundaria.

11.switch(opc2) {

12. Submenú de sabor.

13.case 2:

14. Ingreso manual.

15.while(getchar() != '\n');

16. Limpia buffer.

17.puts("Ingresa el sabor: ");

18. Solicita el sabor.

19.gets(Registro.key.sabor);

20. Guarda el sabor ingresado en la estructura Registro.

21.break;

22. Fin del caso 2 (manual).

23.case 1:

24. Lista predefinida.

25.do {

26. Ciclo do-while para validar la elección.

27.puts("|| 1: Fresa || 2: Chocolate || 3: Limón ||");

28. Lista de sabores de paleta.

29.sccanf("%d", &opc3);

30. Opción elegida de la lista.

31.switch(opc3) {

32. Selección de sabor por lista.

33.case 1:

34. Fresa.

35.strcpy(Registro.key.sabor, "Fresa");

36. Guarda "Fresa" en el registro.

37.break;

38. Fin del case 1.

39.case 2:

40. Chocolate.

41.strcpy(Registro.key.sabor, "Chocolate");

42. Guarda "Chocolate".

43.break;

44. Fin del case 2.

45.case 3:

46. Limón.

```
1.strcpy(Registro.key.sabor, "Limón");
2. Guarda "Limón".
3.break;
4. Fin del case 3.
5.default:
6. Opción inválida.
7.puts("Opcion invalida");
8. Mensaje de error.
9.break;
10. Fin del default.
11.} while(opc3 != 1 && opc3 != 2 && opc3 != 3);
12. Repite mientras la opción sea inválida.
13.break;
14. Fin del case 1 de sabor (lista).
```

## Ingreso de precio

- 1.puts("Ingrese el precio del producto: ");
2. Solicita el precio.
- 3.scanf("%d", &Registro.key.precio);
4. Guarda el precio en el registro.

## Guardar venta en archivo

- 1.fwrite(&Registro, sizeof(producto), 1, binario);
2. Escribe la venta en el archivo ventas.dat.
- 3.co++;
4. Incrementa el contador de ventas.
- 5.printf("\nVenta registrada exitosamente\n");
6. Mensaje de confirmación.

## Pregunta para otra venta

- 1.puts("Desea ingresar otra venta?");
2. Pregunta al usuario si quiere registrar otra venta.
- 3.puts("|| 1: Si || 2: No ||");
4. Opciones de continuar o terminar.
- 5.scanf("%d", &opc);
6. Lee la opción.

## Cierre del ciclo y archivo

- 1.} while(opc == 1);
2. Repite todo el proceso si el usuario elige 1 (sí).
- 3.fclose(binario);
4. Cierra el archivo ventas.dat.
- 5.return co;
6. Devuelve la cantidad total de ventas registradas en la sesión.
- 7.}
8. Fin de la función vantaface().

```

640     int cancelar(producto Registro, int co){
641         //abres archivos
642         FILE *fac, *temporal;
643         factura F;
644         int pos=0,facturab;
645
646         if( (fac = fopen("facturas.dat","rb"))==NULL){
647             printf("Error al abrir el archivo");
648             return 0;
649         }
650
651
652         temporal=fopen("t.dat","wb");
653         if (temporal == NULL) {
654             printf("Error al crear el archivo temporal\n");
655             fclose(fac);
656             return 0;
657         }
658
659
660         //buscas la factura que quieras borrar
661         printf("Ingresa el ID o clave de la factura que buscar para eliminar: \n ");
662         scanf("%d",&facturab);
663
664         while(fread(&F, sizeof(factura), 1, fac)>0){
665
666             if(facturab != F.clave){
667                 fwrite(&F, sizeof(factura), 1, temporal);
668             }
669         }
670         //aqui se encuentra la factura
671         else{
672             pos=1;
673             printf("Factura clave %d: eliminada exitosamente...\n", F.clave);
674         }

```

```

670         //aqui se encuentra la factura
671         else{
672             pos=1;
673             printf("Factura clave %d: eliminada exitosamente...\n", F.clave);
674         }
675     }
676     //cierro archivos
677     fclose(fac);
678     fclose(temporal);
679
680     //si no se encontro el ticket o factura
681     if(!pos)
682         printf("No se encontro la factura \n");
683     else {
684         remove("facturas.dat");
685         rename("t.dat", "facturas.dat");
686         remove("t.dat");
687         return 1;
688     }
689 }
690 }
```

## Inicio de la función

- 1.int cancelar() {
2. Inicio de la función. Esta función permite cancelar un registro de venta o producto del archivo, eliminándolo físicamente.
- 3.FILE \*binario, \*temp;
  - binario → archivo original donde están los registros.
  - temp → archivo temporal donde se guardarán los registros que no se cancelen.
- 4.producto Registro;
5. Variable de tipo producto que sirve para leer cada registro del archivo.
- 6.int clave, co = 0, encontrado = 0;
7. Variables:
  - clave → clave o ID del registro que se desea cancelar.
  - co → contador de registros procesados.
  - encontrado → bandera para saber si se encontró el registro.

## Abrir archivos

- 1.binario = fopen("ventas.dat", "rb");
2. Abre el archivo original en modo lectura binaria.
- 3.if(binario == NULL) {
4. Verifica si hubo error al abrir el archivo.
- 5.puts("Error al abrir el archivo");
6. Mensaje de error.
- 7.return 0;
8. Termina la función si hay error.
- 9.}
10. Fin del bloque if.
- 11.temp = fopen("temp.dat", "wb");
12. Abre un archivo temporal para guardar los registros que no serán cancelados.

## Solicitud de clave a cancelar

- 1.puts("Ingrese la clave del registro a cancelar: ");
2. Solicita al usuario la clave del registro.
- 3.scnf("%d", &clave);
4. Lee la clave.

## Recorrido del archivo original

- 1.while(fread(&Registro, sizeof(producto), 1, binario)) {
2. Ciclo que lee cada registro del archivo original.
- 3.if(Registro.key.id != clave) {
4. Si el registro no coincide con la clave a cancelar:
- 5.fwrite(&Registro, sizeof(producto), 1, temp);
6. Escribe el registro en el archivo temporal.
- 7.co++;
8. Incrementa contador de registros guardados.
- 9.} else {
10. Si la clave coincide (registro a cancelar):
- 11.encontrado = 1;
12. Marca que se encontró el registro a cancelar.
- 13.puts("Registro cancelado");
14. Muestra mensaje de cancelación.
- 15.}
16. Cierre del if-else.
- 17.}
18. Cierre del ciclo while.

## Cerrar archivos

- 1.fclose(binario);
2. Cierra el archivo original.
- 3.fclose(temp);
4. Cierra el archivo temporal.

## Reemplazar archivo original

- 1.remove("ventas.dat");
2. Elimina el archivo original.
- 3.rename("temp.dat", "ventas.dat");
4. Renombra el archivo temporal como archivo original, dejando solo los registros que no fueron cancelados.
- 5.return co;
6. Devuelve el número de registros que quedaron en el archivo después de la cancelación.
- 7.}
8. Fin de la función cancelar().