# Assignment Description

The program will assist a ship that sells silicone. The program will get information about the shape, size, weight, and price. Then, it will take the orders from the user. Finally, the program will display the total cost of the order plus the shipping cost.

# GitHub URL (optional)

[**https://github.com/CarlosLiza16/CSCI-101/blob/main/M07%20Programming\_2.cpp**](https://github.com/CarlosLiza16/CSCI-101/blob/main/M07%20Programming_2.cpp)

# Readme Documentation

Input Information: The user will be prompt to enter basic information like name, last name, address. Then a list will be displayed on the screen. The user must choose one of the options form the list.

* Choose an option from the list

1. Pendant M 17.10 48
2. Dragon L 108.76 200
3. Heart S 26.68 22
4. Ornament M 33.52 101
5. Seahorse L 33.52 111
6. Domino L 54.04 145

//user will enter an option

Output Information: If the input is correctly entered by the user as shown above. Then the user will be asked to add a new order, if not the user will be prompt with a message error.

* That is not a valid input try again (if user does not enter a number)
* That is not a number from the list (if the user enters a number that is not in the list)
* You have reached the maximum orders (if the user wants to enter more than 10 orders)

# Source Code of All files (.h, .cpp)

1. /\* Program name: M07\_programming assignment2.cpp
2. \*  Author: Carlos Lizarazu
3. \*  Date last updated: 10/07/24
4. \*  Purpose: The program will open a file that has a list of items.
5. Then the program will ask information to the user, and then track the order that they are making.
6. \*/
7. #include <fstream>
8. #include <iomanip>
9. #include <iostream>
10. #include <limits>
11. #include <string>
12. #include <vector>
13. using namespace std;
14. // Ceclarinf the enum for the size
15. enum sizeType {
16. S, // small
17. M, // medium
18. L  // large
19. };
20. // Declaring the struct for the mold
21. struct moldType {
22. string shape;
23. sizeType size;
24. double price;
25. double weight;
26. };
27. // declaring the struct for the customer
28. struct customerType {
29. string fname;
30. string lname;
31. string address;
32. };
33. // declaring the order struct
34. struct orderType {
35. customerType customer;
36. vector<moldType> products; // array or vector for the molds
37. int numProdOrdered;
38. double subTotal;
39. double totalWeight;
40. double shippingCost;
41. };
42. // ReadCatalog function 1
43. void readCatalog(vector<moldType> &productCatalog, int &numProducts) {
44. ifstream file("D:/Ivy Tech/Fall Semester 2024/CSCI-101/C++/M07 Assignments/catalog.txt"); // opening the file
45. moldType mold;
46. char sizeChar;
47. // clear any precioud data
48. productCatalog.clear();
49. // Read each line, and storage in the vector
50. while (file >> mold.shape >> sizeChar >> mold.price >> mold.weight) {
51. // identify the size and map it
52. if (sizeChar == 'S') {
53. mold.size = S;
54. } else if (sizeChar == 'M') {
55. mold.size = M;
56. } else if (sizeChar == 'L') {
57. mold.size = L;
58. }
59. productCatalog.push\_back(mold);
60. }
61. numProducts = productCatalog.size();
62. file.close();
63. }
64. //Function 3 print the product
65. void printProduct(ostream& os, const moldType& mold){
66. string sizeStr;
67. switch (mold.size){
68. case S:sizeStr = "Small";break;
69. case M: sizeStr = "Medium"; break;
70. case L: sizeStr = "Large"; break;
71. }
72. os <<mold.shape<<" - "<<sizeStr<<" $"<<fixed<<setprecision(2)<<mold.price<<" "<<mold.weight<<"g"<<endl;
73. }
74. //Function 4 print order
75. void printOrder(ostream& os, const orderType& order){
76. os << " "<< endl;
77. os << "Costumer: " << order.customer.fname <<" "<< order.customer.lname << endl;
78. os << "Address: " << order.customer.address << endl;
79. os<<" "<<endl;
80. os << "Products Ordered:"<<endl;
81. for (const auto& product : order.products) {
82. printProduct(os, product);  // Reuse the printProduct function
83. }
84. double totalCost = order.subTotal + order.shippingCost;
86. os <<" "<<endl;
87. os << "Subtotal: $" << fixed << setprecision(2)<< order.subTotal << endl;
88. os << "Total Weight: " << fixed << setprecision(2) << order.totalWeight << "g"<< endl;
89. os << "Shipping Cost: $" << fixed << setprecision(2) << order.shippingCost << endl;
90. os << "Total Cost: $" << fixed << setprecision(2) << totalCost << endl;
91. }
92. //Function 2 get the order from the user
93. void enterOrder(orderType &order, const vector<moldType> &catalog, int numItemInCatalog) {
94. char answer = 'y';
95. int count = 1;
96. int choice;
97. // Get customer information
98. cout << "Enter the costumer first name: ";
99. // using the struct order that also connects with the customer struct
100. cin >> order.customer.fname;
101. cout << "Enter the customer last name: ";
102. cin >> order.customer.lname;
103. cout << "Enter the customer street address: ";
104. cin.ignore();
105. getline(cin, order.customer.address);
106. // The order starts
107. while (answer == 'y') {
108. if (count <= 10) {
109. cout << "Please choose a product from the list: " << endl;
110. // Show the list of the items
111. for (size\_t i = 0; i < numItemInCatalog; ++i) {
112. cout << i + 1 << ": ";
113. printProduct(cout, catalog[i]);  // Use printProduct to display catalog
114. }
115. // Loop until a valid product choice is made
116. bool validChoice = false;
117. while (!validChoice) {
118. //Entering the option from the list
119. if (cin >> choice) {
120. if (choice >= 1 && choice <= numItemInCatalog) {
121. // Valid choice
122. validChoice = true;
123. order.products.push\_back(catalog[choice - 1]);
124. order.subTotal += catalog[choice - 1].price;
125. order.totalWeight += catalog[choice - 1].weight;
126. order.shippingCost = 0.58 \* (order.totalWeight / 28.35); // Calculate shipping cost
127. count += 1;
128. }else {
129. // Invalid number, out of range
130. cout << "That is not a valid choice. Please choose a product from the list above." << endl;
131. }
132. }else{
133. cout << "You entered something that is not a number. Please try again." << endl;
134. cin.clear(); // Clear the error state
135. cin.ignore(numeric\_limits<streamsize>::max(), '\n');
136. }
137. }
138. //display that order
139. for (const auto &product : order.products) {
140. cout << "Item: " << product.shape << " - $" << product.price << " "
141. << product.weight << "g added." << endl;
142. }
144. // Ask if they want to add another product
145. cout << "Do you want to enter another product? [y]es - [n]o: ";
146. cin >> answer;
147. cin.ignore(numeric\_limits<streamsize>::max(), '\n');
148. cout<<" "<<endl;
149. }else{
150. cout << "You have reached the limit of 10 products per order." << endl;
151. break;
152. }
153. }
154. }
155. int main() {
156. vector<moldType> productCatalog;
157. int numProducts;
158. orderType order;
159. // Initialize the order fields
160. order.subTotal = 0.0;
161. order.totalWeight = 0.0;
162. order.shippingCost = 0.0;
163. // Read the catalog from the file and storage it in the arrar/vec
164. readCatalog(productCatalog, numProducts);
165. // Initializing the order
166. cout << "Welcome to Judy's Resin Molds!" << endl;
167. enterOrder(order, productCatalog, numProducts);
168. //print the order
169. printOrder(cout,order);
171. return 0;
172. }

# Three Use Case Screen Shots

A screenshot of a computer screen

Description automatically generated A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated A screen shot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated A screen shot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated A screenshot of a computer screen

Description automatically generated