# Assignment Description

Judy runs a shop that sells molds used in arts and crafts, specifically for use with resin. The molds are made of silicone and come in various shapes and sizes.  Each mold weighs a different amount which affects the shipping cost.  Judy needs a program to help her process orders for her molds.  You will create a program that allows Judy to enter an order for a mold and create an invoice in the form of an output file.

# GitHub URL (optional)

<https://github.com/wesleyhixon/Programming-Assignments/tree/a0b90821b63035efaee14a3ab9d9b74b3dc95972/M07%20Programming%20Assignment%202>

# Readme Documentation

Input Information: Input is customer name, address, and products to order

Output Information: Output is the product catalog, along with a generated invoice using customer information along with order and pricing.

# Flowchart Screen Shots (optional)

Screen shot(s) here

# UML and Use Case Diagrams (optional)

Screen shot(s) here

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

1. #include <iostream>
2. #include <fstream>
3. #include <string>
4. #include <sstream>
5. #include <iomanip>
6. using namespace std;
7. /\*
8. Program Name: Resin Molds
9. Author: Wesley Hixon
10. Date Last Updated: 7/18/2024
11. Purpose: Generate invoices for resin mold orders
12. \*/
13. enum sizeType { S, M, L };
14. struct moldType{
15. string shape;
16. sizeType size;
17. double price;
18. double weight;
19. };
20. struct customerType{
21. string firstName;
22. string lastName;
23. string streetAddress;
24. string cityStateZip;
25. };
26. struct orderType{
27. customerType customer;
28. moldType products[10];
29. int numProdOrdered;
30. double subTotal;
31. double totalWeight;
32. double shippingCost;
33. };
34. void readCatalog(moldType productCatalog[], int numProducts);                   // Reads catalog.txt for the current catalog (it changes from time to time)
35. void enterOrder(orderType& order, moldType catalog[], int numProducts);         // Prompts the user for their order. Can enter up to 10 products
36. void printProduct(ostream& output, moldType mold);                              // Takes info about a mold and outputs it to ostream provided
37. void printOrder(ostream& output, orderType& order);                             // Takes info about an order and outputs it to ostream provided
38. string getSizeString(sizeType size);                                            // Converts size enum to string
39. bool continuePrompt(string prompt);                                             // Asks if user would like to continue given a prompt
40. int main(){
41. const int numProducts = 20;
42. orderType orders[99];
43. int numOrders = 0;
44. moldType productCatalog[numProducts];
45. readCatalog(productCatalog, numProducts);                                   // Getting catalog from file
46. cout << "Welcome to Judy's Resin Molds!" << endl;                           // Warm greeting
47. bool running = true;
48. while(running){
49. numOrders++;
50. int orderIndex = numOrders - 1;                                         // Incrementing numOrders and getting the index of current order
51. ofstream outputFile;
53. enterOrder(orders[orderIndex], productCatalog, numProducts);            // Getting new order
54. printOrder(cout, orders[orderIndex]);                                   // Printing new order to console
55. string filename = orders[orderIndex].customer.firstName + ".txt";       // Getting filename from customer first name
56. outputFile.open(filename);
57. printOrder(outputFile, orders[orderIndex]);                             // Outputting order to file
58. outputFile.close();
59. running = continuePrompt("Would you like to add another order? ");      // Asking if user would like to make another order
60. }
61. cout << endl << "Have a nice day!";                                         // A polite farewell
62. return 0;
63. }
64. // This function reads the catalog from the file and puts it in the productCatalog[] array
65. void readCatalog(moldType productCatalog[], int numProducts){
66. string fileName = "catalog.txt";
67. ifstream catalogFile;
68. catalogFile.open(fileName);                 // Opening file
69. for(int i = 0; i < numProducts; i++){       // Iterating through each line
70. string line;
71. char size;
72. getline(catalogFile, line);             // getline() and create a stringstream
73. stringstream ss(line);
74. ss >> productCatalog[i].shape;          // Get shape
75. ss >> size;                             // Get size
76. productCatalog[i].size = static\_cast<sizeType>(size);   // Static\_cast char to sizeType
77. ss >> productCatalog[i].price;          // Get price
78. ss >> productCatalog[i].weight;         // Get weight
79. }
80. catalogFile.close();
81. }
82. void enterOrder(orderType& order, moldType catalog[], int numProducts){
83. cout << "Enter the customer first name: ";                          // Get first name
84. cin >> order.customer.firstName;
86. cout << endl << "Enter the customer last name: ";                   // Get last name
87. cin >> order.customer.lastName;
88. cout << endl << "Enter the customer street address: ";              // Get address
89. cin.ignore();
90. getline(cin, order.customer.streetAddress);
91. cout << endl << "Enter the customer city, state, and zip code: ";   // Get city state and zip
92. getline(cin, order.customer.cityStateZip);
93. bool addingProducts = true;
94. while(addingProducts && order.numProdOrdered < 10){                 // While items in order < 10
95. order.numProdOrdered++;                                         // Increment num of products ordered
97. cout << "Please choose a product from the list:" << endl;
99. for(int i = 0; i < numProducts; i++){
100. cout << i + 1 << ". ";
101. printProduct(cout, catalog[i]);                             // Outputting the catalog
102. cout << endl;
103. }
104. int userChoice;
106. bool valid = false;                                             // Get user's choice
107. while(!valid){
108. cin >> userChoice;
109. if(!cin || userChoice < 0 || userChoice > numProducts){
110. cout << "Invalid input. Please enter a number between 1 and " << numProducts << "." << endl;
111. cin.clear();
112. cin.ignore();
113. }
114. else{
115. valid = true;
116. }
117. }
118. userChoice--;                                                   // Decrementing userChoice to use for array index
120. int currentProductIndex = order.numProdOrdered - 1;             // Getting index for current product in order
121. order.products[currentProductIndex] = catalog[userChoice];      // Saving user's choice in order.products[]
122. moldType& product = order.products[currentProductIndex];        // Making reference for product for readability
123. cout << "Item: ";
124. printProduct(cout, product);                                    // Outputting product added
125. cout << " added."<< endl;
126. order.subTotal += product.price;                                // Adding to subtotal and weight
127. order.totalWeight += product.weight;
129. addingProducts = continuePrompt("Would you like to add another product to this order? ");   // Prompt to continue
130. }
131. if(order.numProdOrdered == 10){
132. cout << "Maximum number of products ordered." << endl;          // 10 products max per order
133. }
134. order.shippingCost = 0.58 \* (order.totalWeight/28.35);              // Calculating shipping cost
135. }
136. // This function outputs the information about 1 mold
137. void printProduct(ostream& output, moldType mold){
138. output << fixed << setprecision(2) << mold.shape << " - " << getSizeString(mold.size)
139. << " $" << mold.price << " " << mold.weight << "g";
140. }
141. // This function outputs the information about 1 order
142. void printOrder(ostream& output, orderType& order){
143. double totalCost;
144. totalCost = order.shippingCost + order.subTotal;                    // Calculating total cost
145. output << "Customer: " << order.customer.firstName << " " << order.customer.lastName << endl
146. << order.customer.streetAddress << endl                             // Outputting customer info
147. << order.customer.cityStateZip << endl;
148. output << endl << "Products Ordered:" << endl;
149. for(int i = 0; i < order.numProdOrdered; i++){
150. printProduct(output, order.products[i]);                        // Outputting ordered products info
151. output << endl;
152. }
153. output << endl << fixed << setprecision(2) << "Subtotal: $" << order.subTotal << endl;  // Outputting subtotal,
154. output << "Total Weight: " << order.totalWeight << "g" << endl;                         // Weight,
155. output << "Shipping Cost: $" << order.shippingCost << endl;                             // Shipping cost,
156. output << "Total Cost: $" << totalCost << endl;                                         // and total cost
157. }
158. // This function converts the sizeType enum to a string
159. string getSizeString(sizeType size){
160. string sizeString;
161. switch(static\_cast<char>(size)){
162. case 'S':
163. sizeString = "Small";
164. break;
165. case 'M':
166. sizeString = "Medium";
167. break;
168. case 'L':
169. sizeString = "Large";
170. break;
171. }
172. return sizeString;
173. }
174. // This function prompts for a yes or no input and returns a bool
175. bool continuePrompt(string prompt){
176. char choice;
177. bool valid = false;
179. while(!valid){
180. cout << prompt;
181. cin >> choice;
182. tolower(choice);
183. if(!cin){                       // Case of input failure
184. cout << "That is not a valid entry. Please enter y or n." << endl;
185. cin.clear();
186. cin.ignore();
187. }
188. else if(choice == 'y'){         // y = true
189. return true;
190. }
191. else if(choice == 'n'){         // n = false
192. return false;
193. }
194. else{                           // else, try again.
195. cout << "That is not a valid entry. Please enter y or n." << endl;
196. cin.ignore();
197. }
198. }
199. return false;                       // This is here so my compiler doesn't yell at me
200. }

# Three Use Case Screen ShotA screenshot of a computer program Description automatically generateds

A screenshot of a computer

Description automatically generated

A screenshot of a computer

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

Description automatically generated

A screenshot of a computer

Description automatically generated