Fresh-Juice Vending Machine

Hissamuddin Shaikh

CSCI 201

16th of June 2018

St. Cloud State University

**Fresh-Juice Vending Machine**

**Problem Statement**:

Create a Juice Machine, using a cash register and several dispensers, that will exhibit all the product options, processes the customer selection and sells the product

**Input/Output Description:**

Input: The option of which Juice the use wishes to purchase.

The Amount of Money to purchase the selected Juice.

Output: The Juice, and Change (if any).

**Variables:**

* ***Variables for the Cash Register (cashRegisterType) class:***
  + *cashonHand*
    - A private variable constructed to store the value of cash in the register, and the money being depositied.
    - Type: int
  + *cashIn*
    - A variable for a parametrical constructor in the class.
    - Type: int
  + *amountIn*
    - A variable for the public function *acceptAmount*, to store the amount of money that would be entered by the User.
    - Type: int
* ***Variables for the Dispenser (dispenserType) class:***
  + *numberOfItems*
    - A private variable constructed to store the number of items that remain in dispenser (for the Juices).
    - Type: int
  + *cost*
    - A private variable constructed to store the cost of a type of Juice.
    - Type: int
  + *setNoofItems*
    - A variable for a parametrical constructor in the class.
    - Type: int
  + *setCost*
    - A variable for a parametrical constructor in the class.
    - Type: int
* ***Variables for the Main Program (Function):***
  + *CRT*
    - Object to access the class *cashRegisterType*.
  + *AppleJuice*
    - Object to access the class *dispenserType and* initialize the *numberOfItems* and *cost* for the product of Apple Juice.
  + *OrangeJuice*
    - Object to access the class *dispenserType* and initialize the *numberOfItems* and *cost* for the product of Orange Juice.
  + *MangoJuice*
    - Object to access the class *dispenserType and* initialize the *numberOfItems* and *cost* for the product of Mango Juice.
  + *BananaJuice*
    - Object to access the class *dispenserType and* initialize the *numberOfItems* and *cost* for the product of Banana Juice.
  + *Selection*
    - A variable to store the Users choice, for which Juice they wish to purchase.
    - Type: char
  + *A*
    - A variable to store the Users choice for Juice, after it has been through the function *InputErrorControl,* which ensures that the Users input is interpreted correctly by the program.
    - Type: char
  + *MoneyEntered*
    - A variable to hold the amount of money entered by the user, in the function *DispenseProduct,* which is responsible for calculating money input and dispensing the chosen product.
    - Type: int
  + *MoneyNeeded*
    - A variable to hold the amount of money that the user still needs to insert to purchase the product, in the function *DispenseProduct,* which is responsible for calculating money input and dispensing the chosen product.
    - Type: int
  + *MoneyChange*
    - A variable to hold the excess amount of money that the user has inserted, in the function *DispenseProduct,* which is responsible for calculating money input and dispensing the chosen product.
    - Type: int

**Program Design:**

1. Prompt the User to select what kind of Juice they wish to purchase, through a Menu.
2. Process the selection and ensure that the option is interpreted correctly.
3. Prompt the User to insert the Amount of Money required to purchase the desired Juice, by displaying how much the Juice is.
4. Calculate if the Amount of Money inserted is the right amount. If it is less, ask the user to insert more. If the amount is more, then return the excess amount back to the User.
5. Dispense the Product and ask the user if they wish to purchase more. If they enter the designated selection for exiting the program, terminate the program.

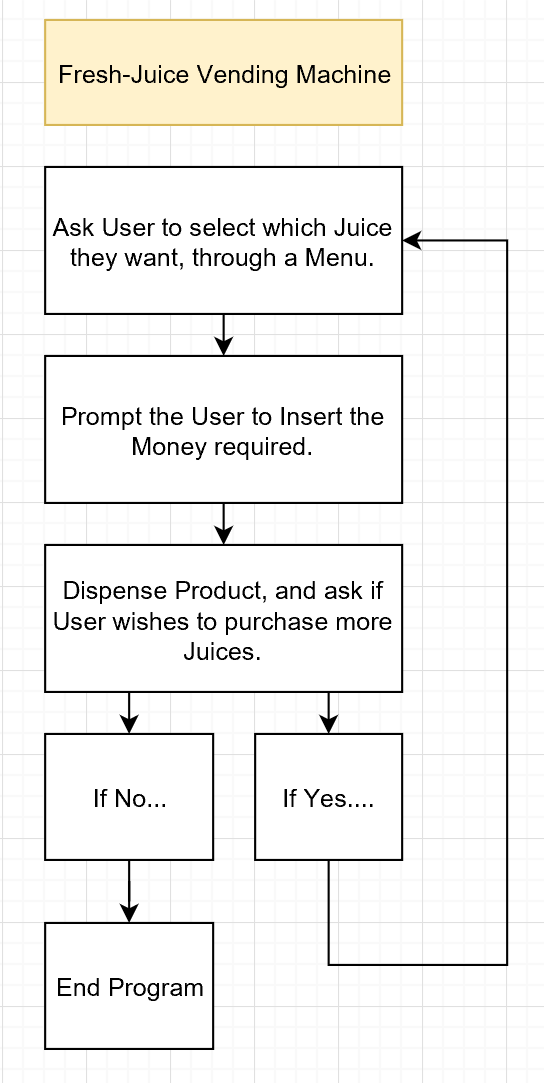
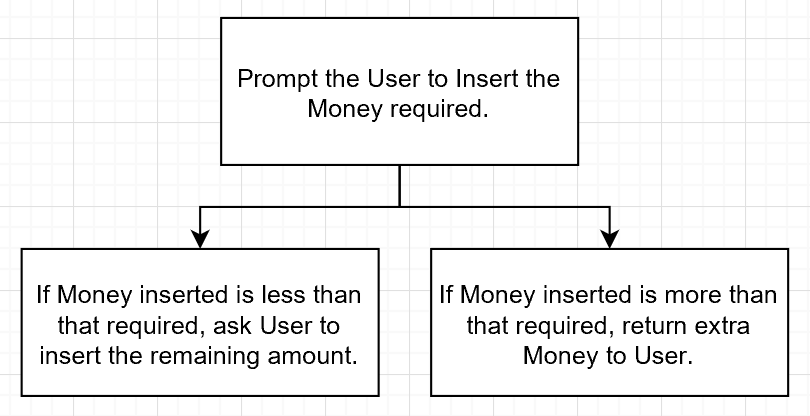
 **Flowchart:**

Figure. A

Note: “Prompt the User to Insert the Money required”, is further explained below in Figure. B.

Figure. B

**Algorithm Development (Detailed Pseudocode):**

* Set *CRT* as the object to access the class *cashRegisterType.*
* Set *AppleJuice(5, 95)* as the object to access the class *dispenserType,* with *5* and *95*, being parameter values for the Constructor (*setNoOfItems* = 5 and *setCost* = 95*)*.
* Set *OrangeJuice(5, 80)* as the object to access the class *dispenserType,* with *5* and *80*, being parameter values for the Constructor (*setNoOfItems* = 5 and *setCost* = 80*)*.
* Set *MangoJuice(5, 75)* as the object to access the class *dispenserType,* with *5* and *75*, being parameter values for the Constructor (*setNoOfItems* = 5 and *setCost* = 75*)*.
* Set *BananaJuice(5, 50)* as the object to access the class *dispenserType,* with *5* and *50*, being parameter values for the Constructor (*setNoOfItems* = 5 and *setCost* = 95*)*.
* Prompt the User to insert their selection, for which Juice they wish to purchase through a Menu, that looks like the following:
  + Fresh-Juice Vending Machine
  + A - Apple Juice
  + B - Orange Juice
  + C - Mango Juice
  + D - Banana Juice
  + E - Exit Program
  + Please Enter A, B, C, D, or E, depending on which Juice you wish to purchase:

Store the Users input in the variable *Selection*.

* Run the information in *Selection*, through the function *InputErrorControl(Selection)*, which will convert lowercase into uppercase (‘a’ into ‘A’, ‘b’ into ‘B’), so that the Program can interpret the choice of the User correctly. Store this in the variable *A*.
* Within a while loop, that is meant to run as long as the variable *A* is not equal to “E”, setup the following to acquire money from the User and dispense the Product.
  + Set up a switch statement that uses the value of *A*, to determine the course of action. The options would be as follows:
    - If the User selected option ‘A’, run the function *DispenseProduct(AppleJuice, CRT).*
    - If the User selected option ‘B’, run the function *DispenseProduct(OrangeJuice, CRT).*
    - If the User selected option ‘C’, run the function *DispenseProduct(MangoJuice, CRT).*
    - If the User selected option ‘D’, run the function *DispenseProduct(BananaJuice, CRT).*
    - Otherwise display “Invalid Selection”, for Default*.*
  + Display the Menu (shown above) again, take the new input and store that in *Selection.* Run this input through *InputErrorContorl(Selection)*, and store it in the variable *A.*
* The function *dispenseProduct()* will do the following:
  + Based on the Juice selected, it will retrieve the value of the *cost*, (through the member function *getCost* of the class *dispenserType)* and display it. It will also prompt the user to insert the Amount of Money that the Juice costs, which is stored in *MoneyEntered*, so the product can be dispensed.
  + If the amount that the User has inserted, which is stored in *MoneyEntered*, is less than that of the *getCost*. A prompt will be administered asking the user to enter the remaining amount (calculated by subtracting the *MoneyEntered* from the *getCost*). This will be stored in *MoneyNeeded*. *MoneyEntered* will then be recalculated by adding the *MoneyEntered* and the *MoneyNeeded*.
  + If the amount that the User has inserted, which is stored in *MoneyEntered*, is exactly equal to the g*etCost*, then make the sale (through *makeSale,* a member function of the class *dispenserType* which will decrement the stock of that type of juice by 1) and deposit the money, stored in *MoneyEntered*, through *acceptAmount* (a member function of the class *cashRegisterType*). A prompt will then tell the User to collect their product from the bottom of the Vending Machine.
  + If the amount that the User has inserted, which is stored in *MoneyEntered* is more than that of the *getCost*. The change will be calculated by subtracting the *getCost* from the *MoneyEntered* and stored in *MoneyChange*. Moreover, the value being passed on to the *acceptAmount* is that of *getCost* as that is how much the user has technically paid for and will then make the sale (through *makeSale*). A prompt will then tell the User to collect their product from the bottom of the Vending Machine, and their change of (*MoneyChange*) from the Change Compartment.

**Program Listing:-**

***dispenserType.h:***

1. class dispenserType
2. {
3. public:
4. /\*Function to show the Number of Items, present in the Machine.\*/
5. int getNoOfItems() const;
6. /\*Function to retrieve the cost that has been set.\*/
7. int getCost() const;
8. /\*Function to decrement the stock of an item by 1 on sale\*/
9. void makeSale();
10. /\*Default Constructor\*/
11. dispenserType();
12. /\*Constructor with Parameters\*/
13. dispenserType(int setNoOfItems, int setCost);
14. private:
15. /\*Variable to store the Number of Items in the Dispenser\*/
16. int numberOfItems;
17. /\*Variable to store the Cost of an Item\*/
18. int cost;
19. };

***dispenserType.cpp:***

1. #include <iostream>
2. #include "dispenserType.h"
3. using namespace std;
4. int dispenserType::getNoOfItems() const
5. {
6. return numberOfItems;
7. }
8. int dispenserType::getCost() const
9. {
10. return cost;
11. }
12. void dispenserType::makeSale()
13. {
14. numberOfItems--;
15. }
16. dispenserType::dispenserType()
17. {
18. numberOfItems = 50;
19. cost = 50;
20. }
21. dispenserType::dispenserType(int setNoOfItems, int setCost)
22. {
23. if (setNoOfItems >= 0)
24. numberOfItems = setNoOfItems;
25. else
26. numberOfItems = 50;
27. if (setCost >= 0)
28. cost = setCost;
29. else
30. cost = 50;
31. }

***cashRegisterType.h:***

1. class cashRegisterType
2. {
3. public:
4. /\*Function to retrieve the Amount of Money in the Register\*/
5. int getCurrentBalance() const;
6. /\*Function to add money into the Register\*/
7. void acceptAmount(int amountIn);
8. /\*Default Constructor\*/
9. cashRegisterType();
10. /\*Constructor with Parameters\*/
11. cashRegisterType(int cashIn);
12. private:
13. /\*Variable to store the Cash in the Register\*/
14. int cashOnHand;
15. };

***cashRegisterType.cpp:***

1. #include <iostream>
2. #include "cashRegisterType.h"
3. using namespace std;
4. int cashRegisterType::getCurrentBalance() const
5. {
6. return cashOnHand;
7. }
8. void cashRegisterType::acceptAmount(int amountIn)
9. {
10. cashOnHand = cashOnHand + amountIn;
11. }
12. cashRegisterType::cashRegisterType()
13. {
14. cashOnHand = 500;
15. }
16. cashRegisterType::cashRegisterType(int cashIn)
17. {
18. if (cashIn >= 0)
19. cashOnHand = cashIn;
20. else
21. cashOnHand = 500;
22. }

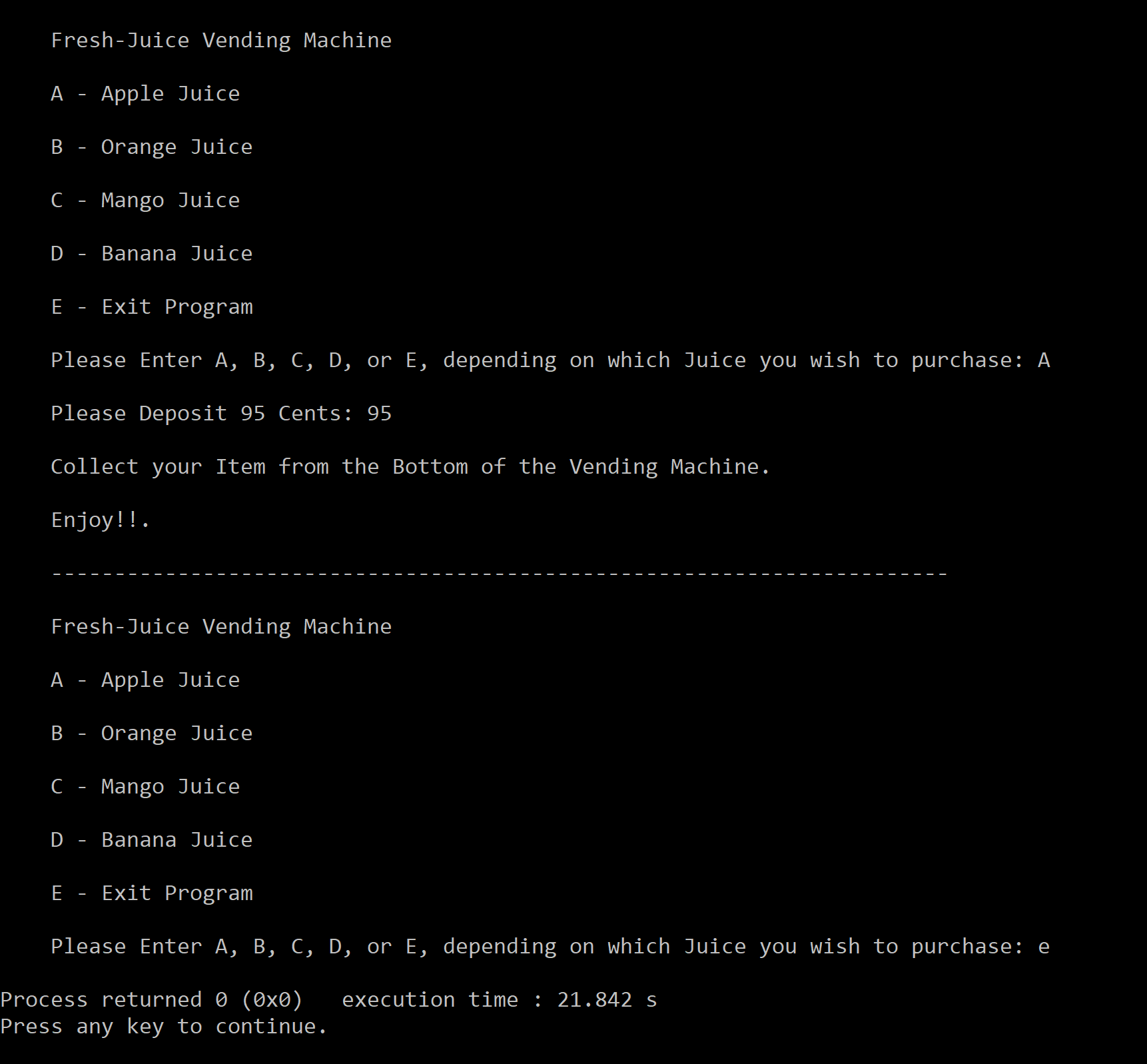
***Main Program:***

1. /\*Two Classes (Juice Machine) by Hissamuddin Shaikh\*/
2. #include <iostream>
3. #include "cashRegisterType.h"
4. #include "dispenserType.h"
5. #include <iomanip>
6. using namespace std;
7. /\*Function Setup(Prototypes)\*/
8. void InterfaceandInputSetup();
9. void DispenseProduct(dispenserType& Product, cashRegisterType& cashDeposited);
10. char InputErrorControl(char);
11. /\*Main Function\*/
12. int main()
13. {
14. /\*Variable and Object Setup\*/
15. cashRegisterType CRT;
16. dispenserType AppleJuice(5,95);
17. dispenserType OrangeJuice(5,80);
18. dispenserType MangoJuice(5,75);
19. dispenserType BananaJuice(5,50);
20. char Selection;
21. char A;
22. /\*Set up for showing the option for Juices\*/
23. InterfaceandInputSetup();
24. cin >> Selection;
25. /\*Error Control for Selection\*/
26. A = InputErrorControl(Selection);
27. /\*Dispensing Setup\*/
28. while (A != 'E')
29. {
30. switch (A)
31. {
32. case 'A':
33. DispenseProduct(AppleJuice, CRT);
34. break;
35. case 'B':
36. DispenseProduct(OrangeJuice, CRT);
37. break;
38. case 'C':
39. DispenseProduct(MangoJuice, CRT);
40. break;
41. case 'D':
42. DispenseProduct(BananaJuice, CRT);
43. break;
44. default:
45. cout << endl << setw(22) << "Invalid selection." << endl;
46. cout << endl << setw(75) << "-----------------------------------------------------------------------" << endl;
47. break;
48. }
49. /\*Setup for the End Condition\*/
50. InterfaceandInputSetup();
51. cin >> Selection;
52. /\*Error Control for Selection\*/
53. A = InputErrorControl(Selection);
54. }
55. return 0;
56. }
57. /\*Setup for the Function to Display the Menu\*/
58. void InterfaceandInputSetup()
59. {
60. cout << endl << setw(31) << "Fresh-Juice Vending Machine" << endl << endl;
61. cout << setw(19) << "A - Apple Juice" << endl << endl;
62. cout << setw(20) << "B - Orange Juice" << endl << endl;
63. cout << setw(19) << "C - Mango Juice" << endl << endl;
64. cout << setw(20) << "D - Banana Juice" << endl << endl;
65. cout << setw(20) << "E - Exit Program" << endl << endl;
66. cout << setw(82) << "Please Enter A, B, C, D, or E, depending on which Juice you wish to purchase: ";
67. }
68. /\*Setup for the Function to make sure that the Program interprets the User Input Correctly\*/
69. char InputErrorControl(char Input)
70. {
71. if (Input == 'a')
72. Input = 'A';
73. else if (Input == 'b')
74. Input = 'B';
75. else if (Input == 'c')
76. Input = 'C';
77. else if (Input == 'd')
78. Input = 'D';
79. else if (Input == 'e')
80. Input = 'E';
81. return Input;
82. }
83. /\*Setup for the Function to take Money in and Dispense Product\*/
84. void DispenseProduct(dispenserType& Product, cashRegisterType& cashDeposited)
85. {
86. int MoneyEntered;
87. int MoneyNeeded;
88. int MoneyChange;
89. if (Product.getNoOfItems() > 0)
90. {
91. cout << endl << setw(19) << "Please Deposit " << Product.getCost()
92. << " Cents: ";
93. cin >> MoneyEntered;
94. if ( MoneyEntered < Product.getCost())
95. {
96. cout << endl <<setw(19) << "Please Deposit "
97. << Product.getCost()- MoneyEntered << " Cents more: ";
98. cin >> MoneyNeeded;
99. MoneyEntered = MoneyEntered + MoneyNeeded;
100. }
101. if ( MoneyEntered < Product.getCost())
102. {
103. cout<< endl << setw(58) <<"\*\*Transaction Terminated, due to Insufficient Funds.\*\*" << endl;
104. }
105. if (MoneyEntered == Product.getCost())
106. {
107. cashDeposited.acceptAmount(MoneyEntered);
108. Product.makeSale();
109. cout << endl << setw(61) << "Collect your Item from the Bottom of the Vending Machine." << endl;
110. cout << endl<< setw(12)<< "Enjoy!!." << endl;
111. }
112. else if (MoneyEntered > Product.getCost())
113. {
114. MoneyChange = MoneyEntered - Product.getCost();
115. cashDeposited.acceptAmount(Product.getCost());
116. Product.makeSale();
117. cout << endl << setw(76)<< "Collect your Item from the Bottom of the Vending Machine, and Change of " <<
118. MoneyChange <<" Cents from the Change Compartment." << endl;
119. cout << endl<< setw(12)<< "Enjoy!!." << endl;
120. }
121. cout << endl << setw(75) << "-----------------------------------------------------------------------" << endl;
122. }
123. else
124. {
125. cout << endl <<setw(33) << "Sorry, this Item is Sold Out." << endl;
126. cout << endl << setw(75) << "-----------------------------------------------------------------------" << endl;
127. }
128. }

**Note:**

Screenshots of the original code are attached at the end of the document (in the notes section), incase this is unclear or something didn’t copy appropriately.

**Sample test run of the program:**

 1st Sample:

2nd Sample:

A screenshot of a cell phone

Description generated with very high confidence

3rd Sample:

A screenshot of a cell phone

Description generated with very high confidence

**Observations, error handling and general comments:**

The program runs correctly as the result in the 1st Sample and 2nd Sample, correspond to manual computations.

**1st Sample (Manual Computation) =**

Selected Option = A (Apple Juice)

Cost = 95

Amount Inserted =95

Appropriate Amount = Yes

Sale Made = Yes

Change Returned (if any) = No

Next Option = e (Exit Program)

This is the same as the one computed by the computer.

**2nd Sample (Manual Computation) =**

Selected Option = b (Orange Juice)

Cost = 80

Amount Inserted = 45

Appropriate Amount = No

Remaining Amount Inserted = 35

Appropriate Amount = Yes

Sale Made = Yes

Change Returned (if any) = No

Next Option = e (Exit Program)

This is the same as the one computed by the computer.

**3rd Sample (Manual Computation) =**

Selected Option =d (Banana Juice)

Cost = 50

Amount Inserted =75

Appropriate Amount = Yes

Sale Made = Yes

Change Returned (if any) = 25

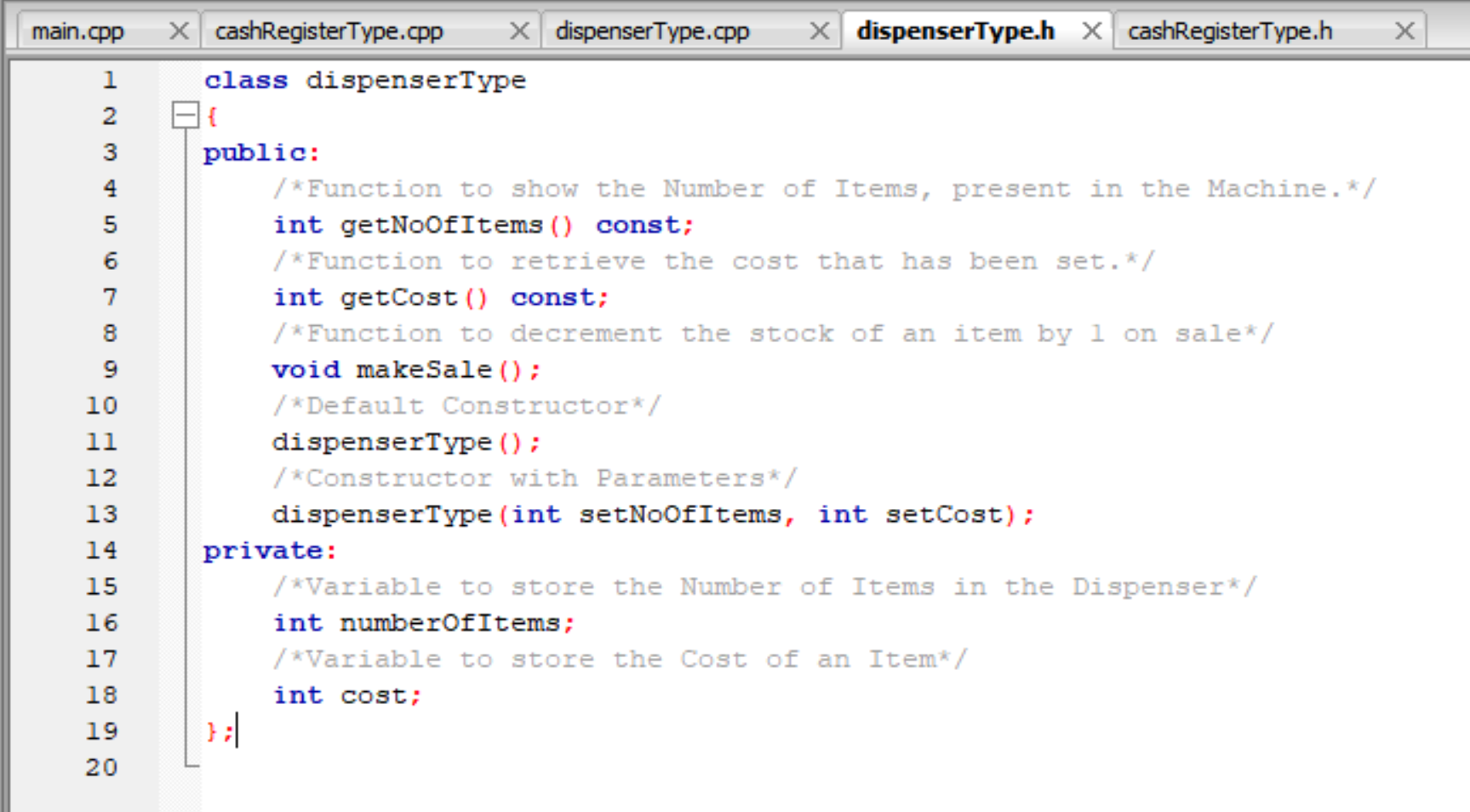
Next Option = e (Exit Program)

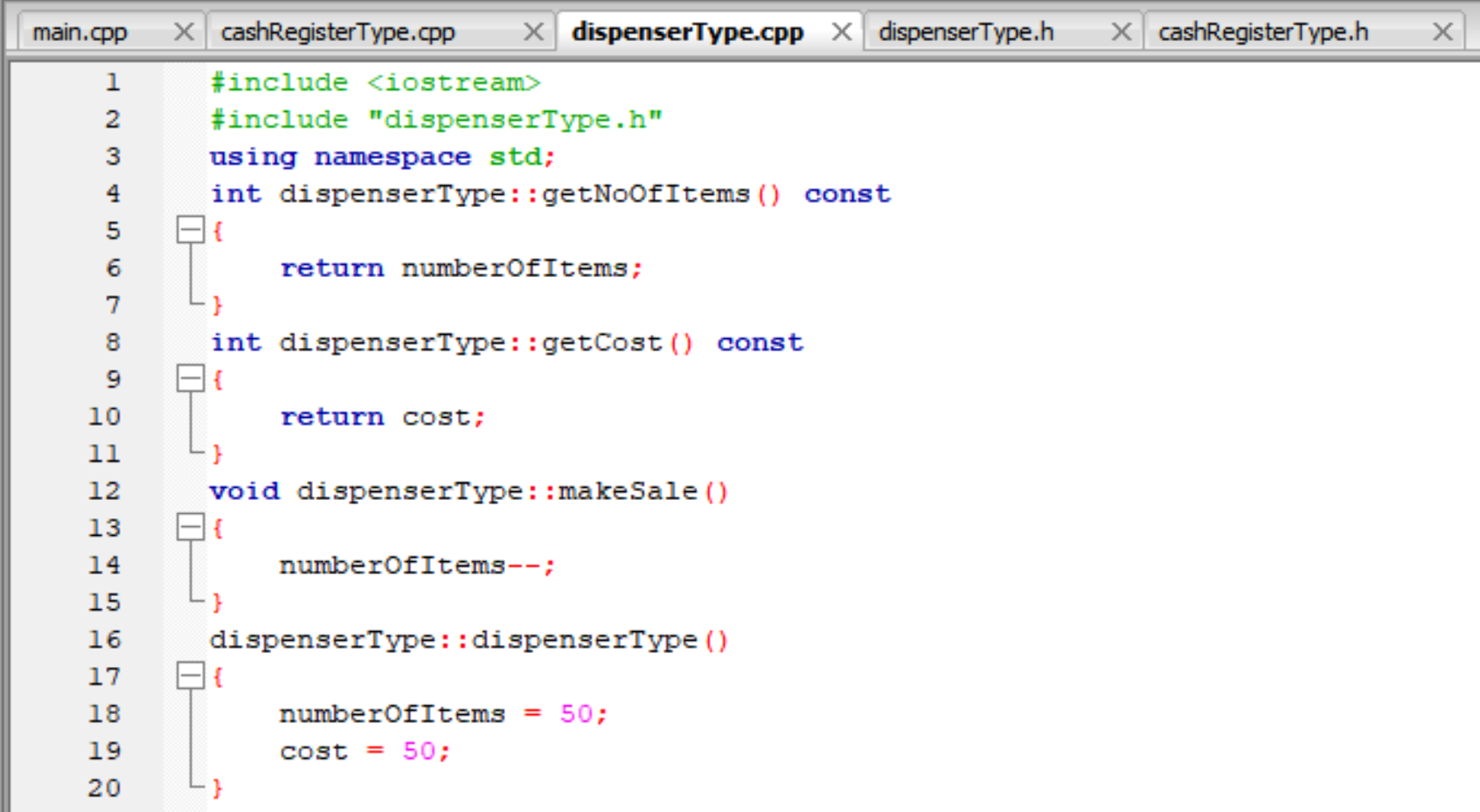
This is the same as the one computed by the computer.

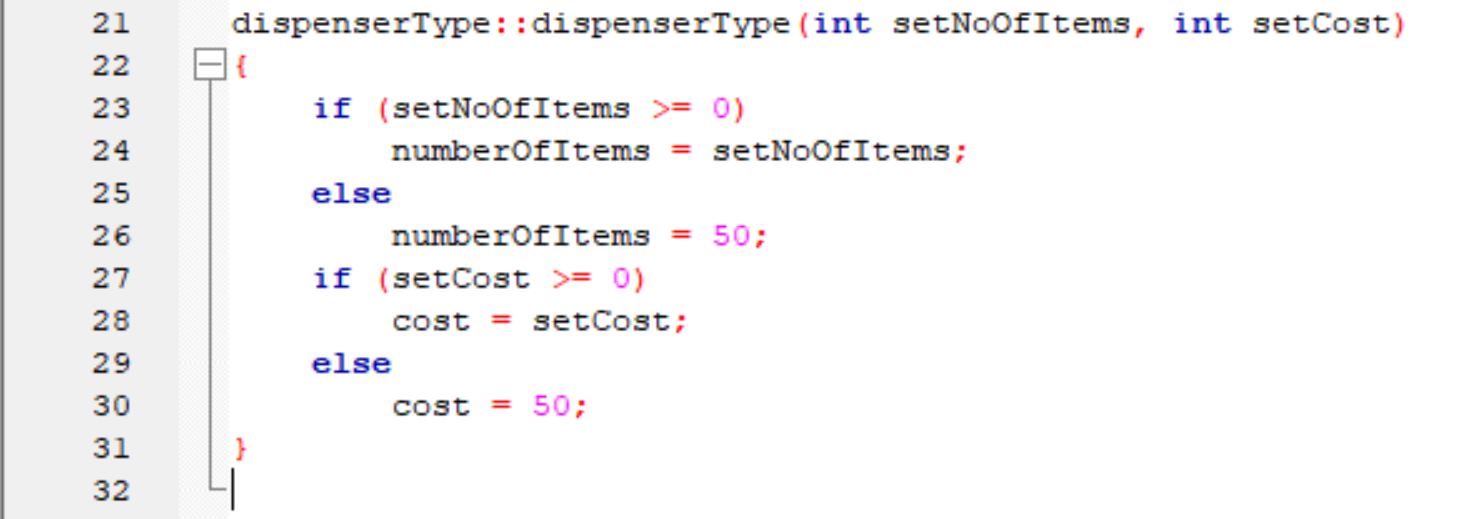
No errors should be encountered, if the user follows the prompts, and enters data in the appropriate format (no decimals, and numbers only where numbers are required). Moreover, various prompts have been setup to limit the input of error causing data, such as the termination of the transaction if the user doesn’t enter the remaining amount correctly, when the first input was less than the cost. Additionally, if the user enters an excess amount on the second prompt for Money, the change is still returned. Furthermore, the user can continue to purchase as many items as he/she likes until the stock runs out (5 for each type of Juice). Therefore, a lot of error control was employed in this program, all of which can not be shown as the report would become excessively long. Not to mention, the presentation of the Program was made as neat and comprehensible as possible.

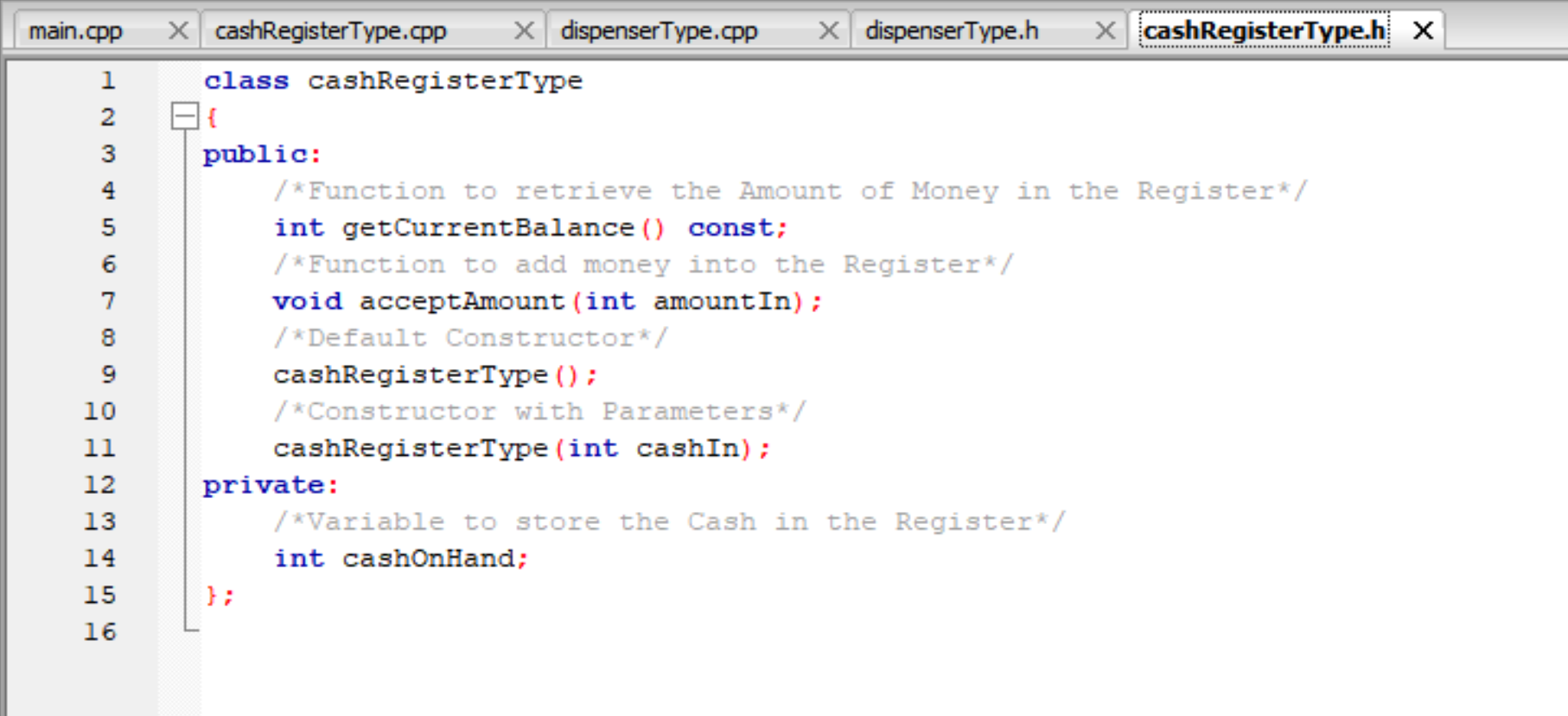
**Conclusions**

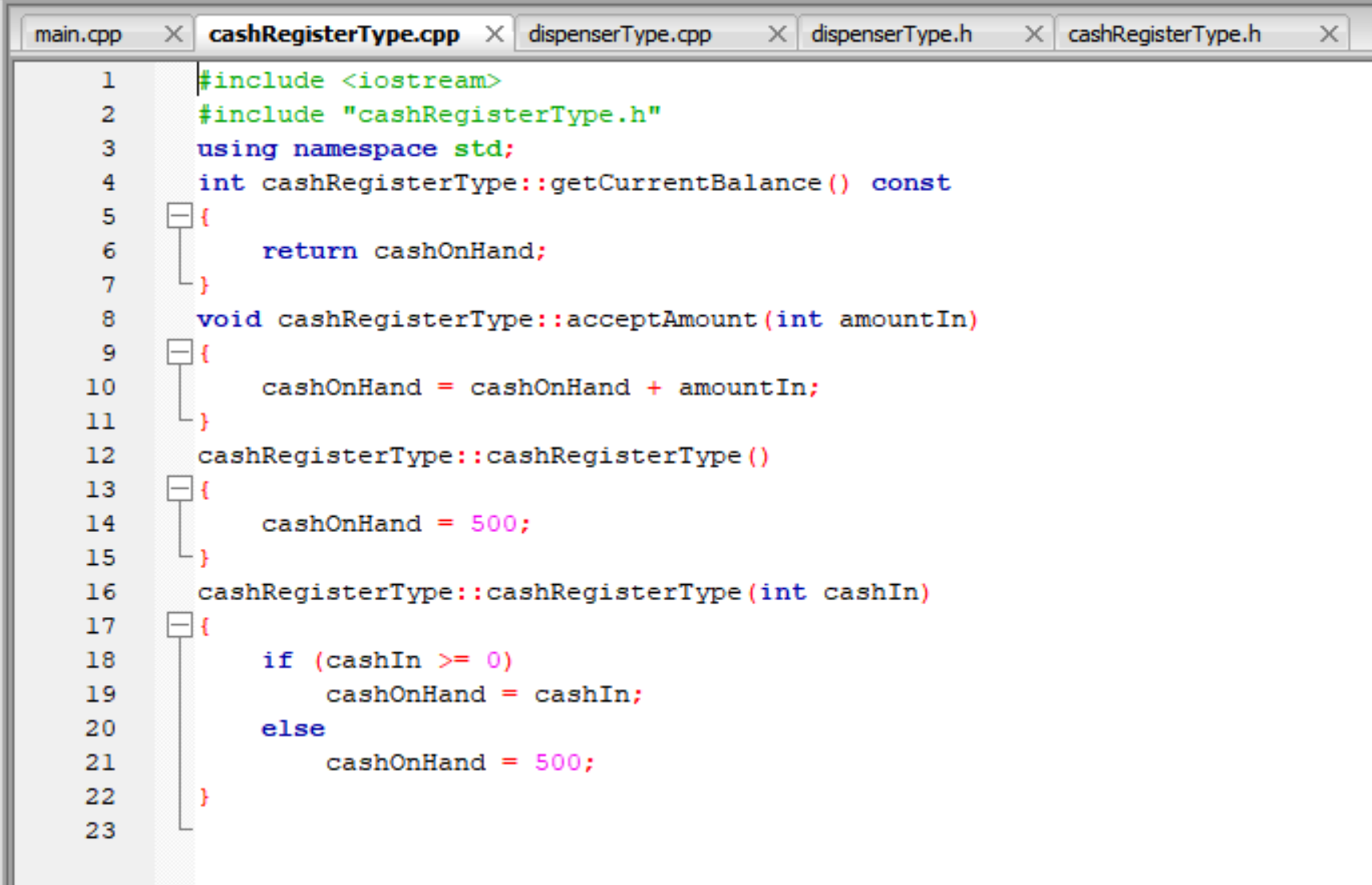
The Fresh-Juice Vending Machine makes great use of the two classes, cashRegisterType and dispenserType, as it constructs a very realistic Vending Machine. I do acknowledge that some of the member functions in these classes were not utilized, but they do function correctly. It’s just that the implementation of all member functions in a single program, is far too difficult and unlikely. Therefore, the Fresh-Juice Vending Machine is a well-rounded program, capable of performing its task effectively.

**Notes:**









A screenshot of a social media post

Description generated with very high confidence

A screenshot of a social media post

Description generated with very high confidence

A screenshot of a social media post

Description generated with very high confidence

A screenshot of a social media post

Description generated with very high confidence

A screenshot of a social media post

Description generated with very high confidence