Table of Contents

Chapter	page no.
1. Introduction	2
2. Main function	3
3. The Product Class	5
a. Variables	5
b. Constructor	5
c. Getters and Setters	6
4. Vending Machine Class	6
a. Variables	7
b. Programming mode functions	7
c. Normal mode Functions	9
d. Maintenance Mode	10
e. Interface	11
5. Test Cases	13

Introduction

Code goal: Vending Machine Software

Programming Language: C++

Requirements

- The system shall simulate a vending machine that sells different beverages like (cola, water, juice)
- The system shall provide Each item with a unique price and a dedicated outlet in the machine
- The system must allow the user to the able to select the required item and deposit the corresponding price through a dedicated slot in the machine After the user deposits the required money the machine must dispense the selected item and return the change if needed
- The system must have two moods (programming, operation)

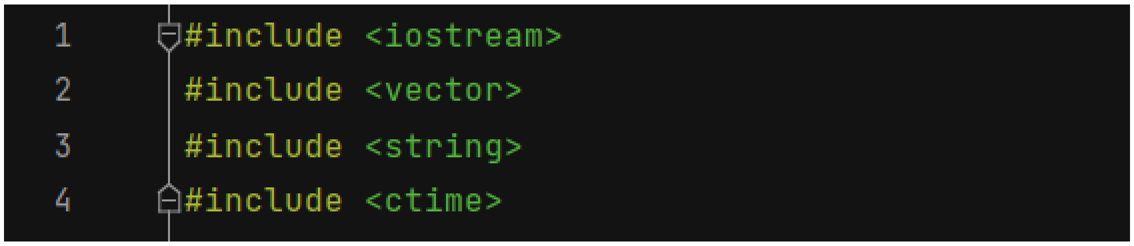
1-<u>In programming mode</u>

The vendor must be able to add different types of products to the machine. The vendor must be able to enter the following information about each product:

1- Name
2- Price
3- Count
4- Expiration date
5- Outlet used to dispense the product

2-<u>In operation mode</u>, the system shall receive a selection a product from the user, dispense the product, deposit the money, and return the change, if there is any.

<u>Libraries</u>



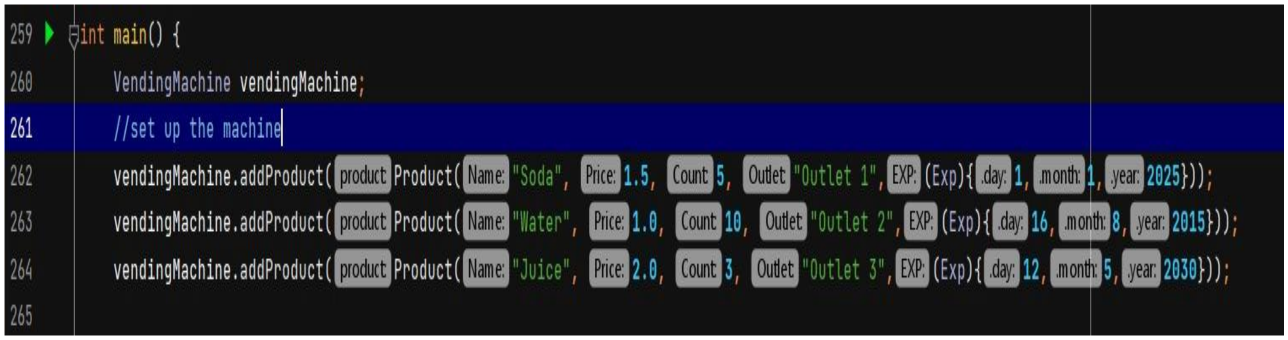
```
#include <iostream>
#include <vector>
#include <string>
#include <ctime>
```

These are the required libraries for the construction of the whole code. We used "***vector***" and "***string***" libraries for some data structures. In addition, the "***ctime***" library is used to access the current time during the implementation.

<u>Main</u>

The vending machine starts with 3 products {soda, water, juice} in three outlets

1. Soda : outlet1 {price :1.5, count:5, Expired date: 1/1/2025}
2. Water: outlet2 {price:1, count:10, Expired date: 16/8/2015}
3. Juice :outlet3 {price:2, count, Expired date: 12/5/2030}



```
int main() {
    VendingMachine vendingMachine;
    //set up the machine
    vendingMachine.addProduct( product Product( Name: "Soda", Price: 1.5, Count: 5, Outlet: "Outlet 1", EXP: (Exp){ .day: 1, .month: 1, .year: 2025}));
    vendingMachine.addProduct( product Product( Name: "Water", Price: 1.0, Count: 10, Outlet: "Outlet 2", EXP: (Exp){ .day: 16, .month: 8, .year: 2015}));
    vendingMachine.addProduct( product Product( Name: "Juice", Price: 2.0, Count: 3, Outlet: "Outlet 3", EXP: (Exp){ .day: 12, .month: 5, .year: 2030}));

```

At the beginning the machine ask for choose between programming mode and user mode.

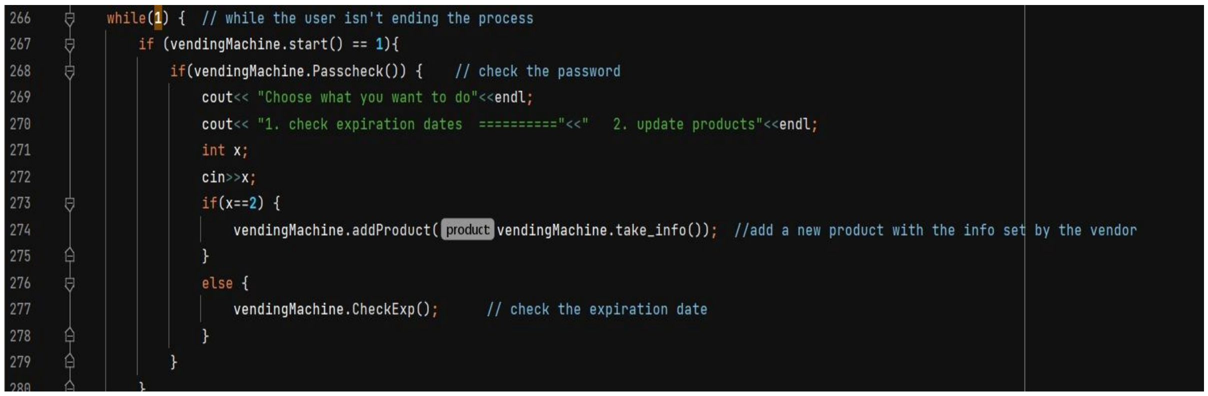
For programming mode press 1 – for user mode press 0

Programming mode

The machine firstly ask for password

If it correct the machine will ask the user to choose what he want to do

1. Check the expired date
2. Add a new product



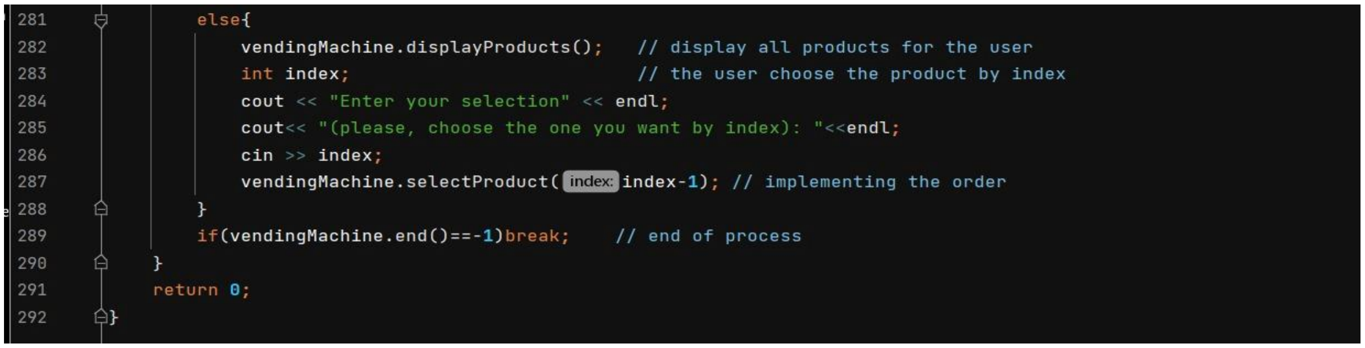
```
    while(1) {  // while the user isn't ending the process
        if (vendingMachine.start() == 1){
            if(vendingMachine.Passcheck()) {    // check the password
                cout<< "Choose what you want to do"<<endl;
                cout<< "1. check expiration dates  =========="<<"   2. update products"<<endl;
                int x;
                cin>>x;
                if(x==2) {
                    vendingMachine.addProduct( product vendingMachine.take_info());  //add a new product with the info set by the vendor
                }
                else {
                    vendingMachine.CheckExp();      // check the expiration date
                }
            }
        }
```

User mode

The machine will ask the user to choose one of the selection

If it available the machine will implement the order

If not the machine will end this process

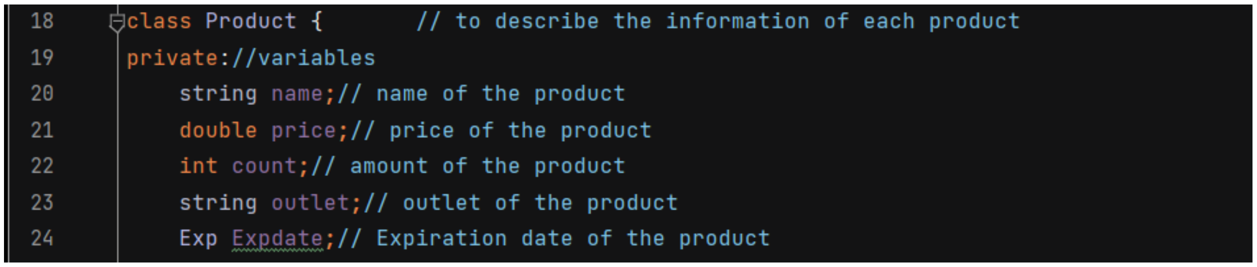


```
        else{
            vendingMachine.displayProducts();   // display all products for the user
            int index;                          // the user choose the product by index
            cout << "Enter your selection" << endl;
            cout<< "(please, choose the one you want by index): "<<endl;
            cin >> index;
            vendingMachine.selectProduct( index: index-1); // implementing the order
        }
        if(vendingMachine.end()==-1)break;     // end of process
    }
    return 0;
}
```

The product class

The product class's purpose is to set, get, and store the information associated with any product in the vending machine.

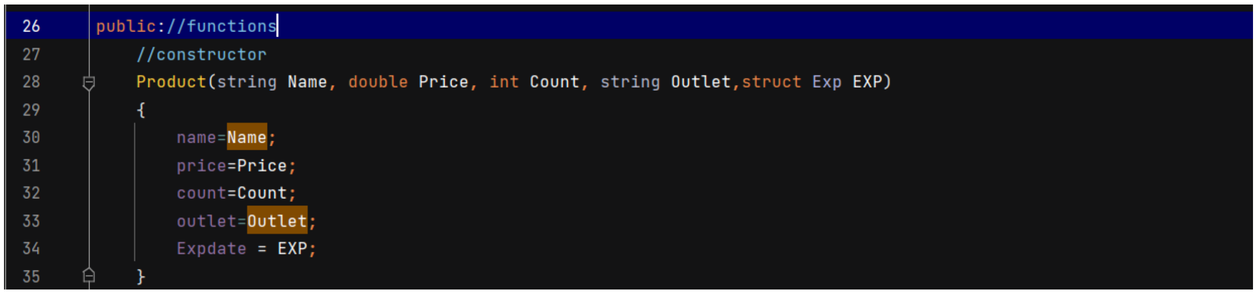
Variables



```
class Product {        // to describe the information of each product
private://variables
    string name;// name of the product
    double price;// price of the product
    int count;// amount of the product
    string outlet;// outlet of the product
    Exp Expdate;// Expiration date of the product
```

We used some variables related to the products' descriptions like name, price, count, outlet, and Expiration date. They are set as private variables to avoid changing them within the class.

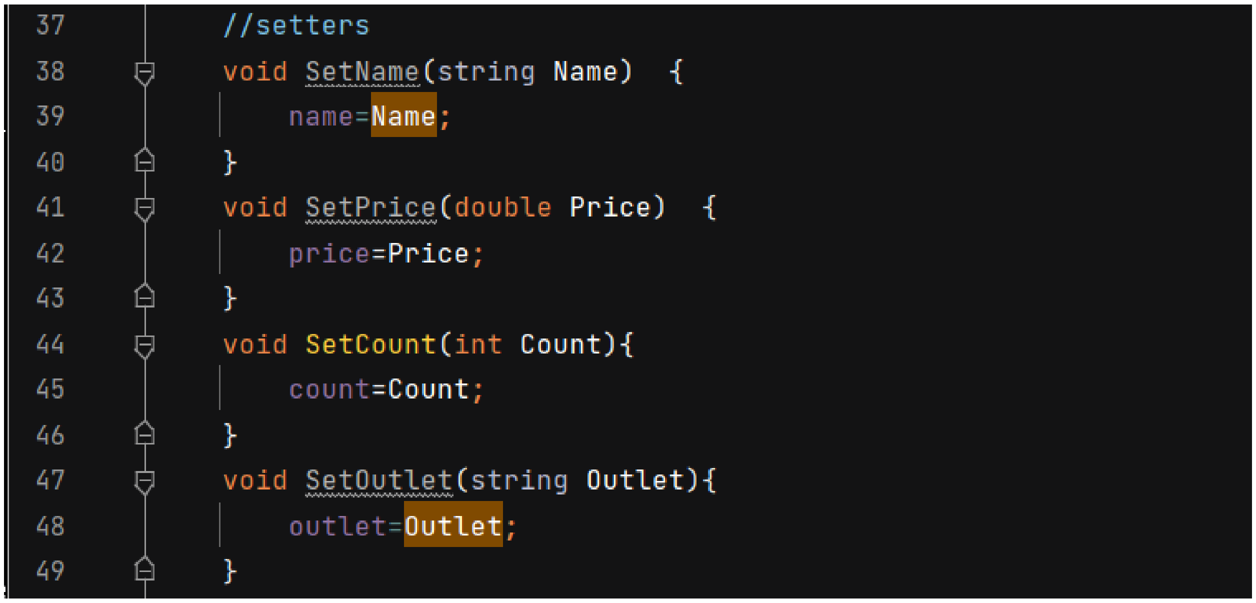
Constructor:



```
public://functions
    //constructor
    Product(string Name, double Price, int Count, string Outlet,struct Exp EXP)
    {
        name=Name;
        price=Price;
        count=Count;
        outlet=Outlet;
        Expdate = EXP;
    }
```

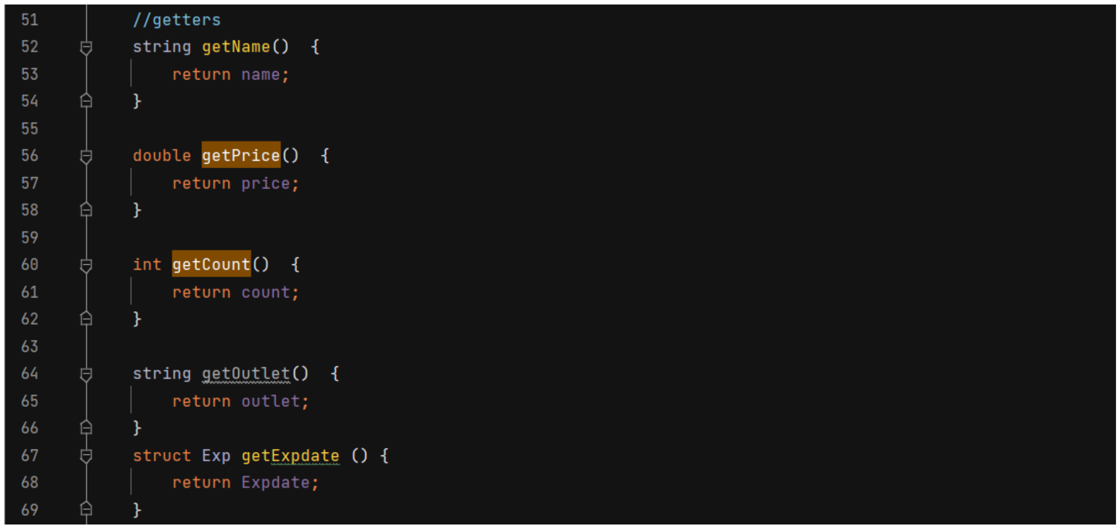
We use the **Product** constructor method to construct the machine's product and automatically assign the values of its details {name, price, count, outlet, exp). All constructors and functions are set as public in the class.

Setters



```
    //setters
    void SetName(string Name)  {
        name=Name;
    }
    void SetPrice(double Price)  {
        price=Price;
    }
    void SetCount(int Count){
        count=Count;
    }
    void SetOutlet(string Outlet){
        outlet=Outlet;
    }
```

Getters



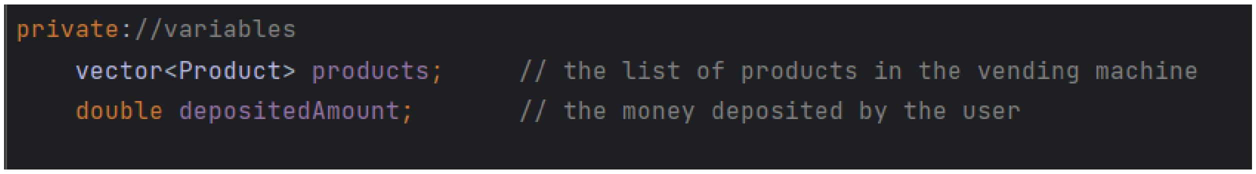
```
//getters
string getName()  {
    return name;
}

double getPrice()  {
    return price;
}

int getCount()  {
    return count;
}

string getOutlet()  {
    return outlet;
}
struct Exp getExpdate () {
    return Expdate;
}
```

Vending Machine Class

In this class, we incorporate the necessary variables and functions essential for the successful operation of the vending machine.

We have two function types, one for the normal mode and the other for the programming one.

Variables

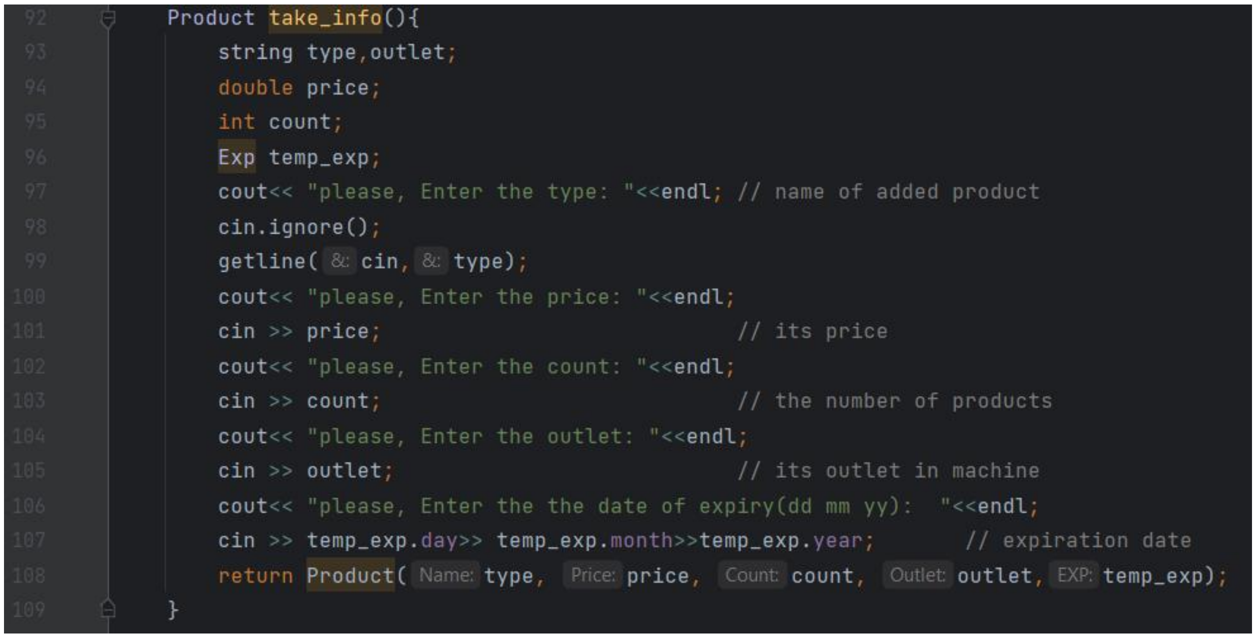


```
private://variables
    vector<Product> products;      // the list of products in the vending machine
    double depositedAmount;        // the money deposited by the user
```

I made the variables private to match the standard. I then made a vector of the class products to hold all the product info and be dynamic in case of any new update. Also, a double holds the user's deposited amount of money.

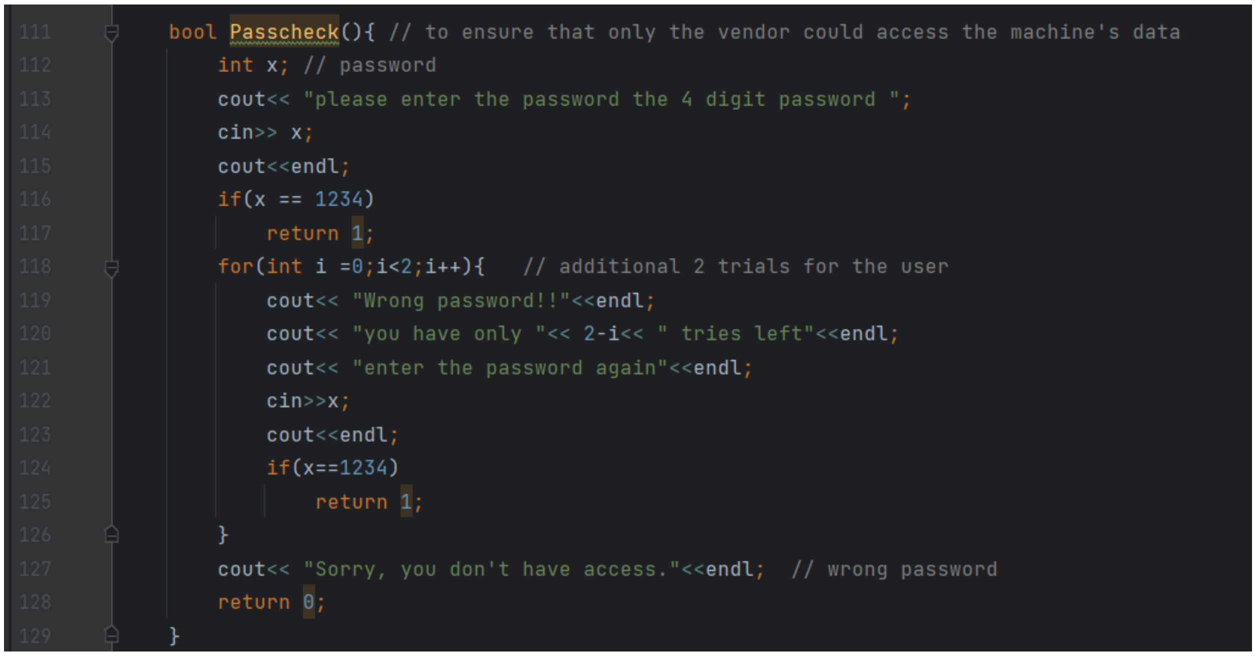
Programming mode functions

Take_Info()


```cpp
Product take_info(){
    string type,outlet;
    double price;
    int count;
    Exp temp_exp;
    cout<< "please, Enter the type: "<<endl; // name of added product
    cin.ignore();
    getline( &: cin, &: type);
    cout<< "please, Enter the price: "<<endl;
    cin >> price;                            // its price
    cout<< "please, Enter the count: "<<endl;
    cin >> count;                            // the number of products
    cout<< "please, Enter the outlet: "<<endl;
    cin >> outlet;                           // its outlet in machine
    cout<< "please, Enter the the date of expiry(dd mm yy):  "<<endl;
    cin >> temp_exp.day>> temp_exp.month>>temp_exp.year;       // expiration date
    return Product( Name: type,  Price: price,  Count: count,  Outlet: outlet,  EXP: temp_exp);
}
```

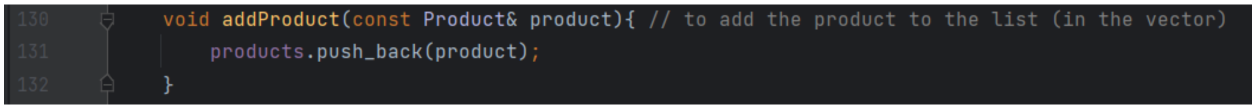
The function receives a string representing the type and outlet, a double for the price (allowing possible fractions), an integer for the count of products, and the expiration date. It then returns the product as an instance of the product class.

Passcheck()


```cpp
bool Passcheck(){ // to ensure that only the vendor could access the machine's data
    int x; // password
    cout<< "please enter the password the 4 digit password ";
    cin>> x;
    cout<<endl;
    if(x == 1234)
        return 1;
    for(int i =0;i<2;i++){   // additional 2 trials for the user
        cout<< "Wrong password!!"<<endl;
        cout<< "you have only "<< 2-i<< " tries left"<<endl;
        cout<< "enter the password again"<<endl;
        cin>>x;
        cout<<endl;
        if(x==1234)
            return 1;
    }
    cout<< "Sorry, you don't have access."<<endl;  // wrong password
    return 0;
}
```

This function receives a password entered by the user as an integer. If the password is correct, the programmer will have permission to access it. If not, the function provides two additional attempts for the programmer to enter the correct password. If the password is still incorrect after these attempts, the function prevents the programmer from making any changes.

addProduct()

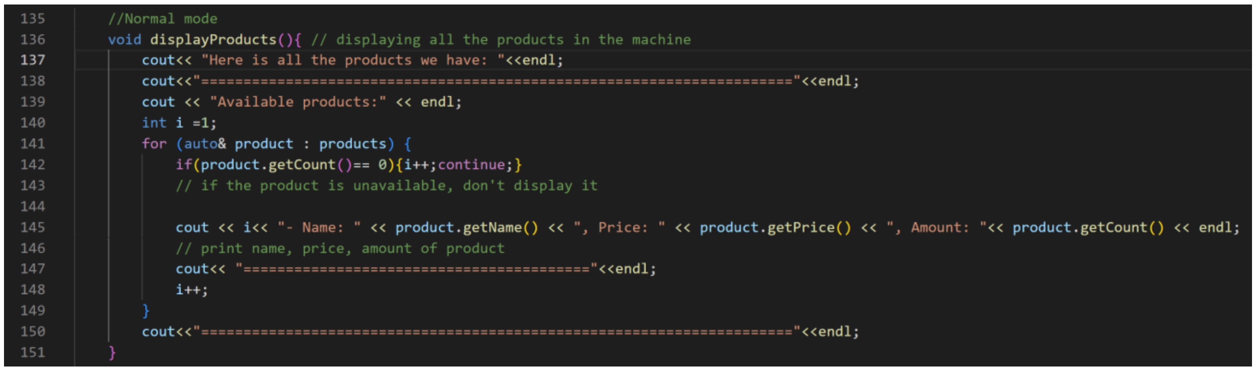


```cpp
void addProduct(const Product& product){ // to add the product to the list (in the vector)
    products.push_back(product);
}
```

This function takes a product and adds it to a vector.

<u>normal mode functions</u>

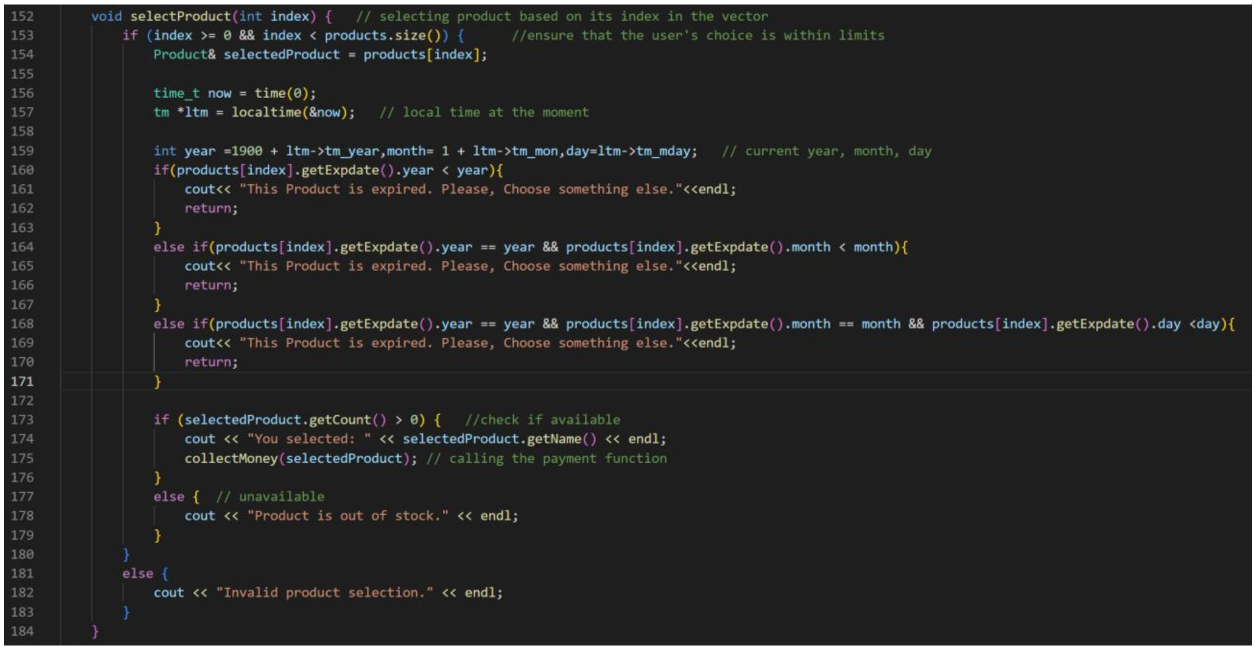
DisplayProducts()



```cpp
//Normal mode
void displayProducts(){ // displaying all the products in the machine
    cout<< "Here is all the products we have: "<<endl;
    cout<<"======================================================================="<<endl;
    cout << "Available products:" << endl;
    int i =1;
    for (auto& product : products) {
        if(product.getCount()== 0){i++;continue;}
        // if the product is unavailable, don't display it

        cout << i<< "- Name: " << product.getName() << ", Price: " << product.getPrice() << ", Amount: "<< product.getCount() << endl;
        // print name, price, amount of product
        cout<< "=========================================="<<endl;
        i++;
    }
    cout<<"======================================================================="<<endl;
}
```

In this function, we display and sort all available product names, prices, and quantities.

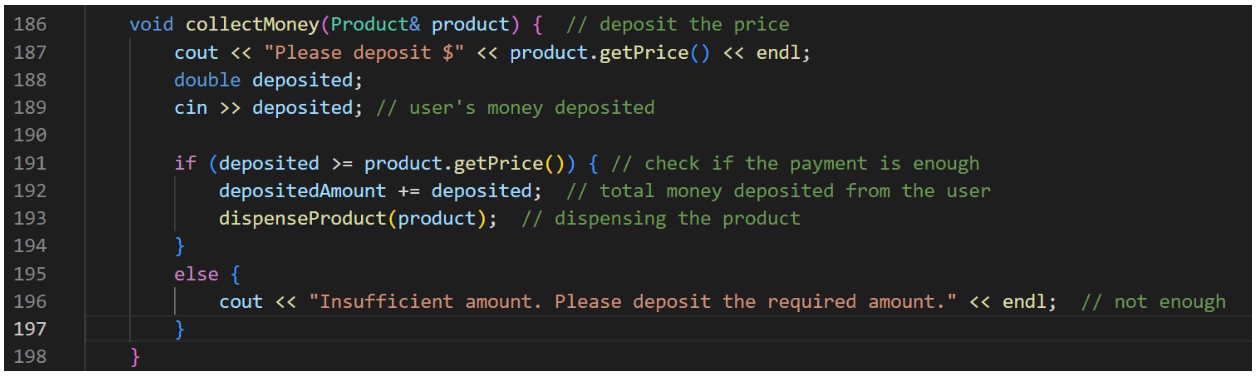
SelectProduct()



```cpp
    void selectProduct(int index) {   // selecting product based on its index in the vector
        if (index >= 0 && index < products.size()) {      //ensure that the user's choice is within limits
            Product& selectedProduct = products[index];

            time_t now = time(0);
            tm *ltm = localtime(&now);   // local time at the moment

            int year =1900 + ltm->tm_year,month= 1 + ltm->tm_mon,day=ltm->tm_mday;   // current year, month, day
            if(products[index].getExpdate().year < year){
                cout<< "This Product is expired. Please, Choose something else."<<endl;
                return;
            }
            else if(products[index].getExpdate().year == year && products[index].getExpdate().month < month){
                cout<< "This Product is expired. Please, Choose something else."<<endl;
                return;
            }
            else if(products[index].getExpdate().year == year && products[index].getExpdate().month == month && products[index].getExpdate().day <day){
                cout<< "This Product is expired. Please, Choose something else."<<endl;
                return;
            }

            if (selectedProduct.getCount() > 0) {   //check if available
                cout << "You selected: " << selectedProduct.getName() << endl;
                collectMoney(selectedProduct); // calling the payment function
            }
            else {  // unavailable
                cout << "Product is out of stock." << endl;
            }
        }
        else {
            cout << "Invalid product selection." << endl;
        }
    }
```

In this function, we ensure that the product is available and has not exceeded its expiration date before displaying it. In the case of either condition not being met, a message will be shown to the user.

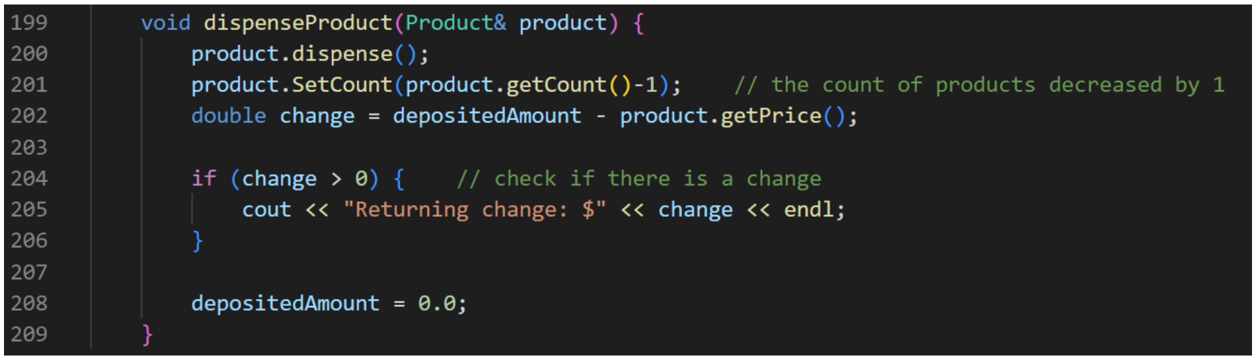
CollectMoney()



```cpp
    void collectMoney(Product& product) {  // deposit the price
        cout << "Please deposit $" << product.getPrice() << endl;
        double deposited;
        cin >> deposited; // user's money deposited

        if (deposited >= product.getPrice()) { // check if the payment is enough
            depositedAmount += deposited;  // total money deposited from the user
            dispenseProduct(product);  // dispensing the product
        }
        else {
            cout << "Insufficient amount. Please deposit the required amount." << endl;  // not enough
        }
    }
```

In this function, we receive money and check if it is sufficient. If not, a message will be displayed to the user.

DispenseProduct()

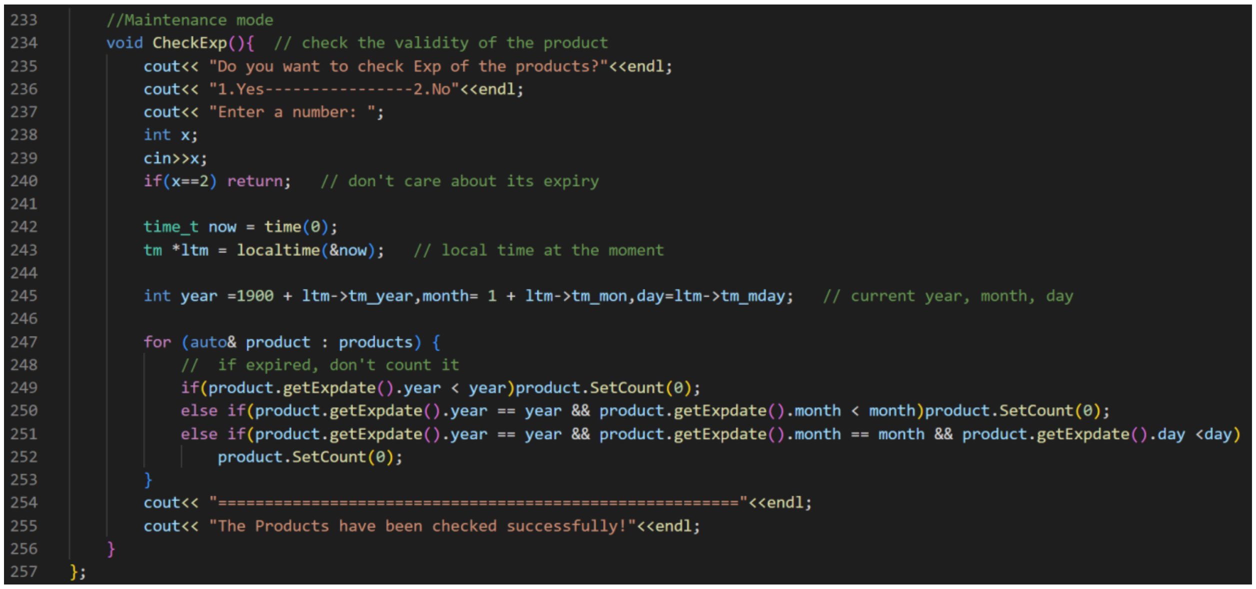


```cpp
    void dispenseProduct(Product& product) {
        product.dispense();
        product.SetCount(product.getCount()-1);    // the count of products decreased by 1
        double change = depositedAmount - product.getPrice();

        if (change > 0) {    // check if there is a change
            cout << "Returning change: $" << change << endl;
        }

        depositedAmount = 0.0;
    }
```

In this function, we retrieve the price of the product, subtract it from the user's money, and if there is a rest money, we provide a message displaying the remaining amount.

Maintenance Mode

CheckExp()

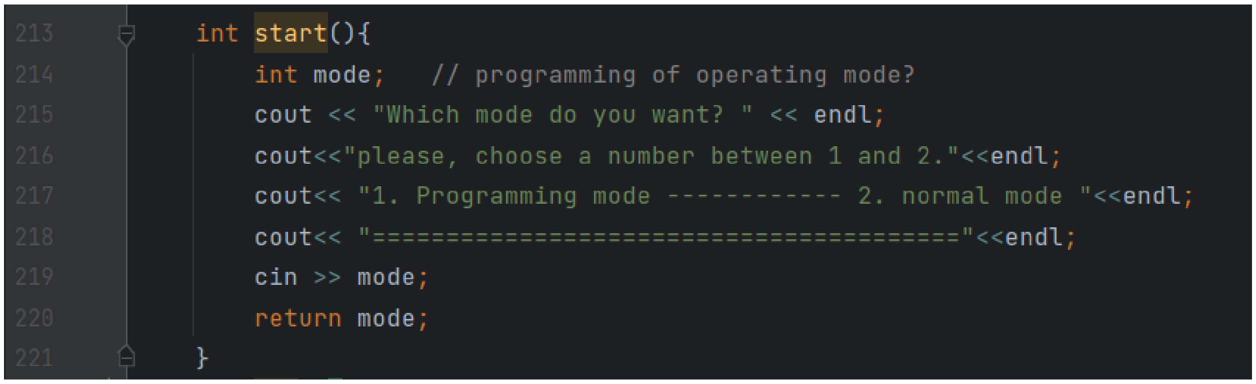


```cpp
    //Maintenance mode
    void CheckExp(){  // check the validity of the product
        cout<< "Do you want to check Exp of the products?"<<endl;
        cout<< "1.Yes---------------2.No"<<endl;
        cout<< "Enter a number: ";
        int x;
        cin>>x;
        if(x==2) return;   // don't care about its expiry

        time_t now = time(0);
        tm *ltm = localtime(&now);   // local time at the moment

        int year =1900 + ltm->tm_year,month= 1 + ltm->tm_mon,day=ltm->tm_mday;   // current year, month, day

        for (auto& product : products) {
            //  if expired, don't count it
            if(product.getExpdate().year < year)product.SetCount(0);
            else if(product.getExpdate().year == year && product.getExpdate().month < month)product.SetCount(0);
            else if(product.getExpdate().year == year && product.getExpdate().month == month && product.getExpdate().day <day)
                product.SetCount(0);
        }
        cout<< "========================================================"<<endl;
        cout<< "The Products have been checked successfully!"<<endl;
    }
};
```

By running this function, it will automatically check all products and leave you a message.

Interface

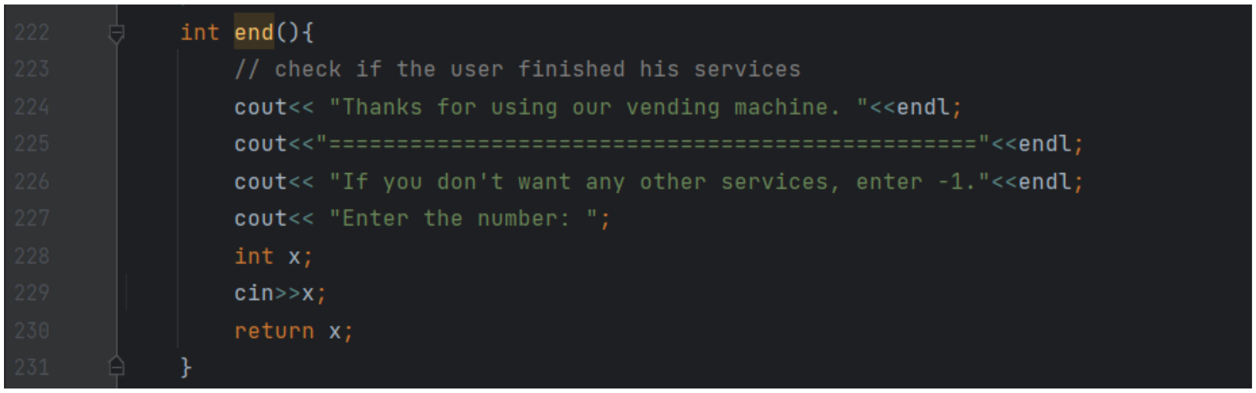
Start()



```
int start(){
    int mode;   // programming of operating mode?
    cout << "Which mode do you want? " << endl;
    cout<<"please, choose a number between 1 and 2."<<endl;
    cout<< "1. Programming mode ------------ 2. normal mode "<<endl;
    cout<< "========================================"<<endl;
    cin >> mode;
    return mode;
}
```

This function prompts the user to choose between two modes and return their choice.

End()

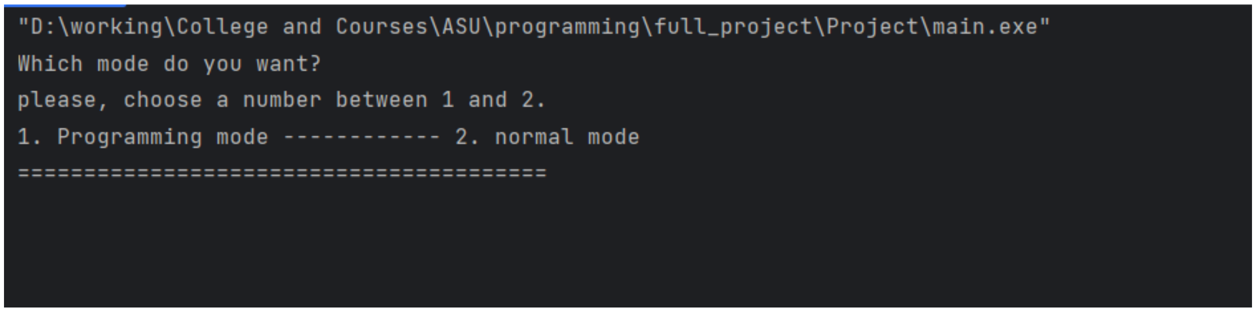


```
int end(){
    // check if the user finished his services
    cout<< "Thanks for using our vending machine. "<<endl;
    cout<<"================================================"<<endl;
    cout<< "If you don't want any other services, enter -1."<<endl;
    cout<< "Enter the number: ";
    int x;
    cin>>x;
    return x;
}
```

This function prompts the user to choose between continuing the service or exiting and return their choice.

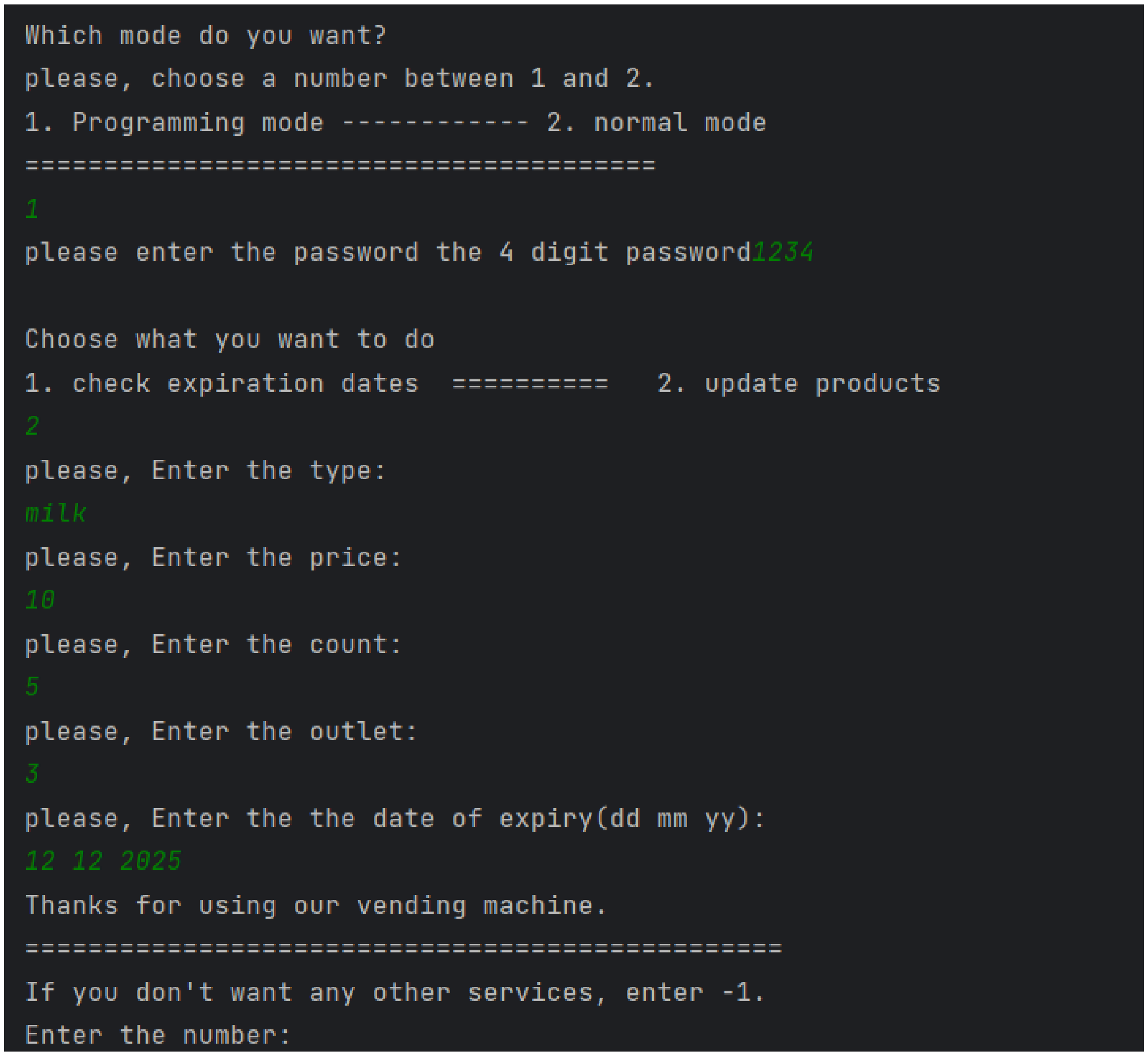
Test Cases

First choose which mode



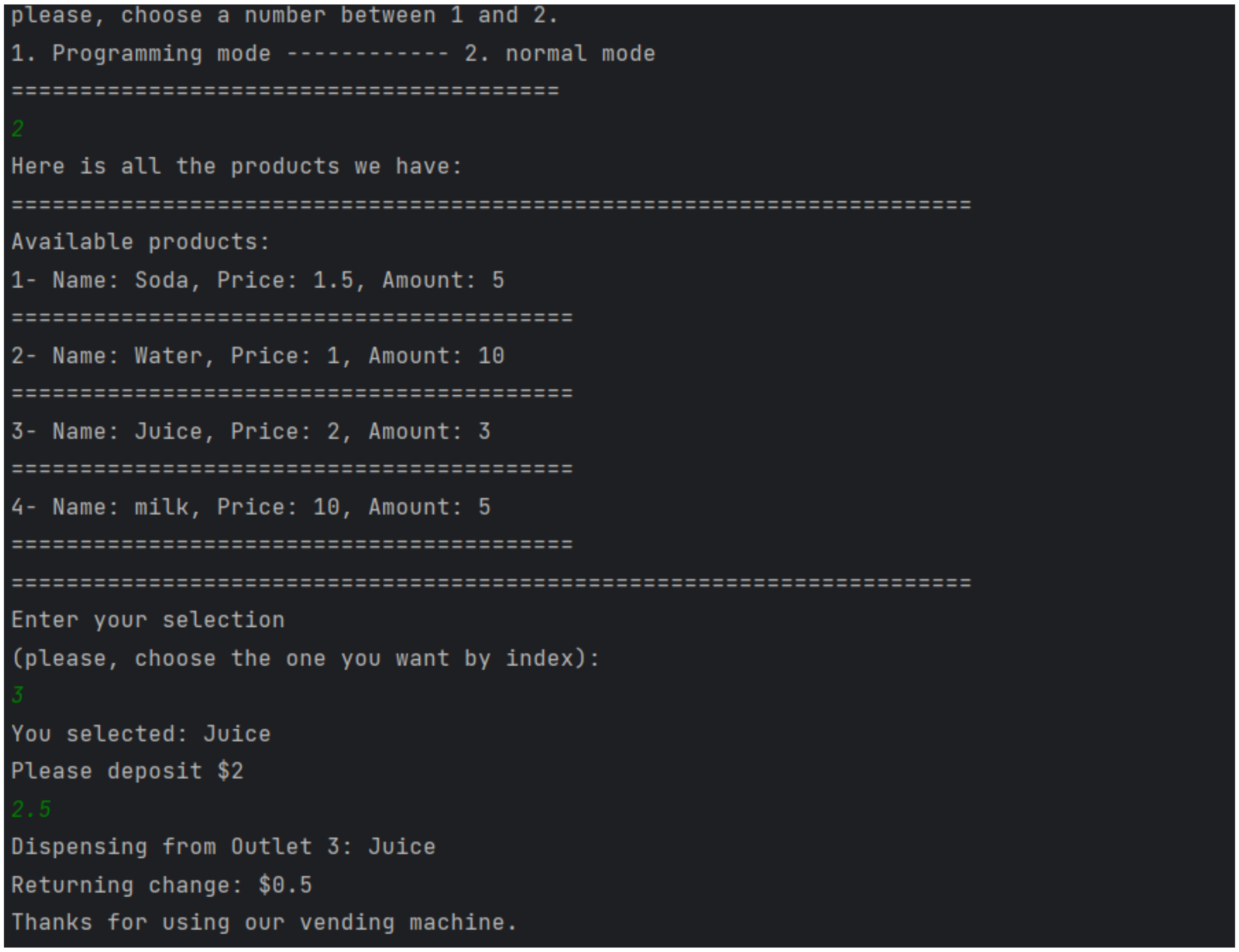
```
"D:\working\College and Courses\ASU\programming\full_project\Project\main.exe"
Which mode do you want?
please, choose a number between 1 and 2.
1. Programming mode ------------ 2. normal mode
========================================
```

If Programming mode and updating

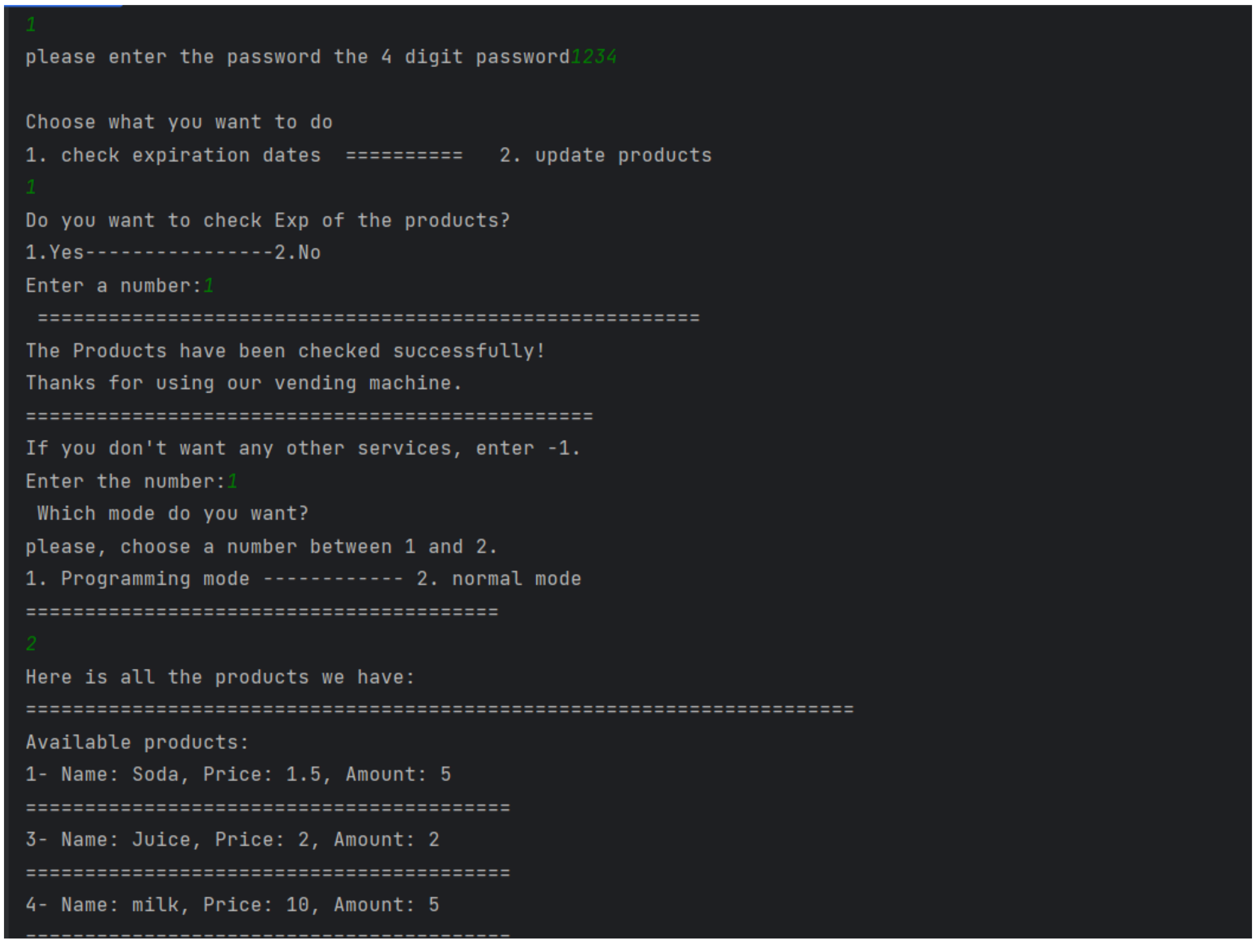


```
Which mode do you want?
please, choose a number between 1 and 2.
1. Programming mode ------------ 2. normal mode
========================================
1
please enter the password the 4 digit password1234

Choose what you want to do
1. check expiration dates  ==========   2. update products
2
please, Enter the type:
milk
please, Enter the price:
10
please, Enter the count:
5
please, Enter the outlet:
3
please, Enter the the date of expiry(dd mm yy):
12 12 2025
Thanks for using our vending machine.
================================================
If you don't want any other services, enter -1.
Enter the number:
```

If other mode



```
please, choose a number between 1 and 2.
1. Programming mode ------------ 2. normal mode
========================================
2
Here is all the products we have:
======================================================================
Available products:
1- Name: Soda, Price: 1.5, Amount: 5
=========================================
2- Name: Water, Price: 1, Amount: 10
=========================================
3- Name: Juice, Price: 2, Amount: 3
=========================================
4- Name: milk, Price: 10, Amount: 5
=========================================
======================================================================
Enter your selection
(please, choose the one you want by index):
3
You selected: Juice
Please deposit $2
2.5
Dispensing from Outlet 3: Juice
Returning change: $0.5
Thanks for using our vending machine.
```



```
1
please enter the password the 4 digit password1234

Choose what you want to do
1. check expiration dates  ==========   2. update products
1
Do you want to check Exp of the products?
1.Yes----------------2.No
Enter a number:1
 =========================================================
The Products have been checked successfully!
Thanks for using our vending machine.
=================================================
If you don't want any other services, enter -1.
Enter the number:1
 Which mode do you want?
please, choose a number between 1 and 2.
1. Programming mode ------------ 2. normal mode
=========================================
2
Here is all the products we have:
=======================================================================
Available products:
1- Name: Soda, Price: 1.5, Amount: 5
==========================================
3- Name: Juice, Price: 2, Amount: 2
==========================================
4- Name: milk, Price: 10, Amount: 5
```