UTM

UNIVERSITI TEKNOLOGI MALAYSIA

Group Name: BOBOBOY

Group Member:

1. AERON GOH MING LUN A22EC0033
2. TAY SHUN WEI A22EC0110
3. LIN CHONG HUI A22EC0184

Course: DATA STRUCTURE AND ALGORITHMS
Section: 04

Lecturer : Ms. LIZAWATI BINTI MI YUSUF

Table of Contents

Objective

The main objective of BOBOBOY's Restaurant Management System (RMS) is to optimize the order management system with effectiveness features and to minimize the time to get through our ordering system. Our system is created for a full process from ordering to payment session and it can be done via through the system and the customers can get the food in the best condition without wasting any time on waiting or getting used to the system.
All the features, panel and functions are to make sure the customers have good feedback to our restaurant and satisfy not only about the food, but also the whole system and environment of the BOBOBOY restaurant.

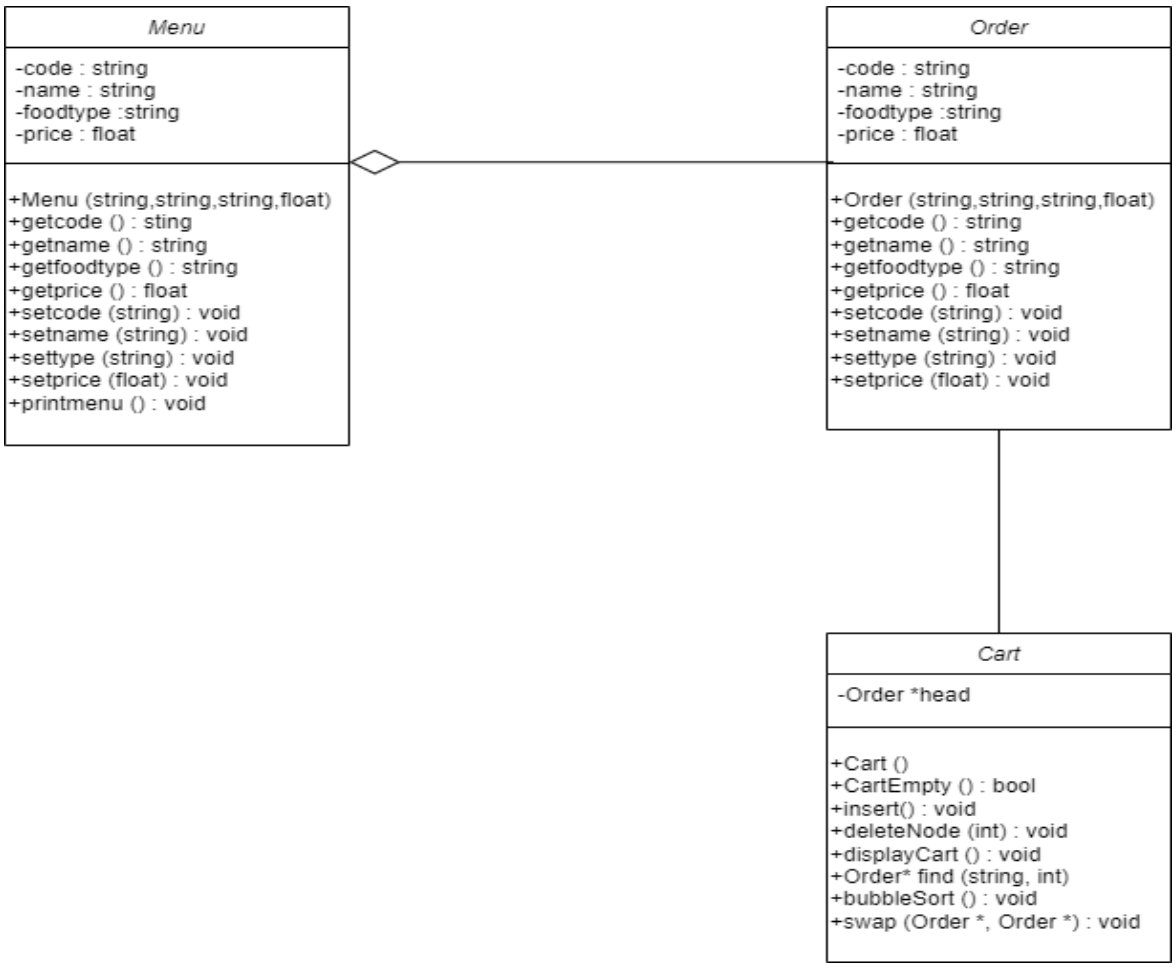
Synopsis

In a restaurant, it is most important that RMS should be user friendly and show a detailed menu in a simple way. Besides, the customer's emotion is very important as they want to get the food they ordered as fast as possible. In BOBOBOY restaurant, the RMS contained two main functions in the menu which is sorting and searching. The purpose of implementing these two functions is because we want the customers to look through the menu based on their favorite arrangement such as food name with ascending alphabet arrangement. . After that, if they cannot find what specific food they want on the menu, they can type the details or info about that food on the search panel and it will show the position of that product so next time they can straight away order from the menu without wasting time. Besides, inside the order cart, the customer can add on or delete the item straight away inside the cart so there is no need to go back to the menu to do this action. Inside the cart also got a searching and sorting function which helps customers to get through their order fast and easier. All of these are to make our customers like the ordering system of BOBOBOY restaurants and they feel easy and simple when ordering their desired food.

Linked List in Ordering System

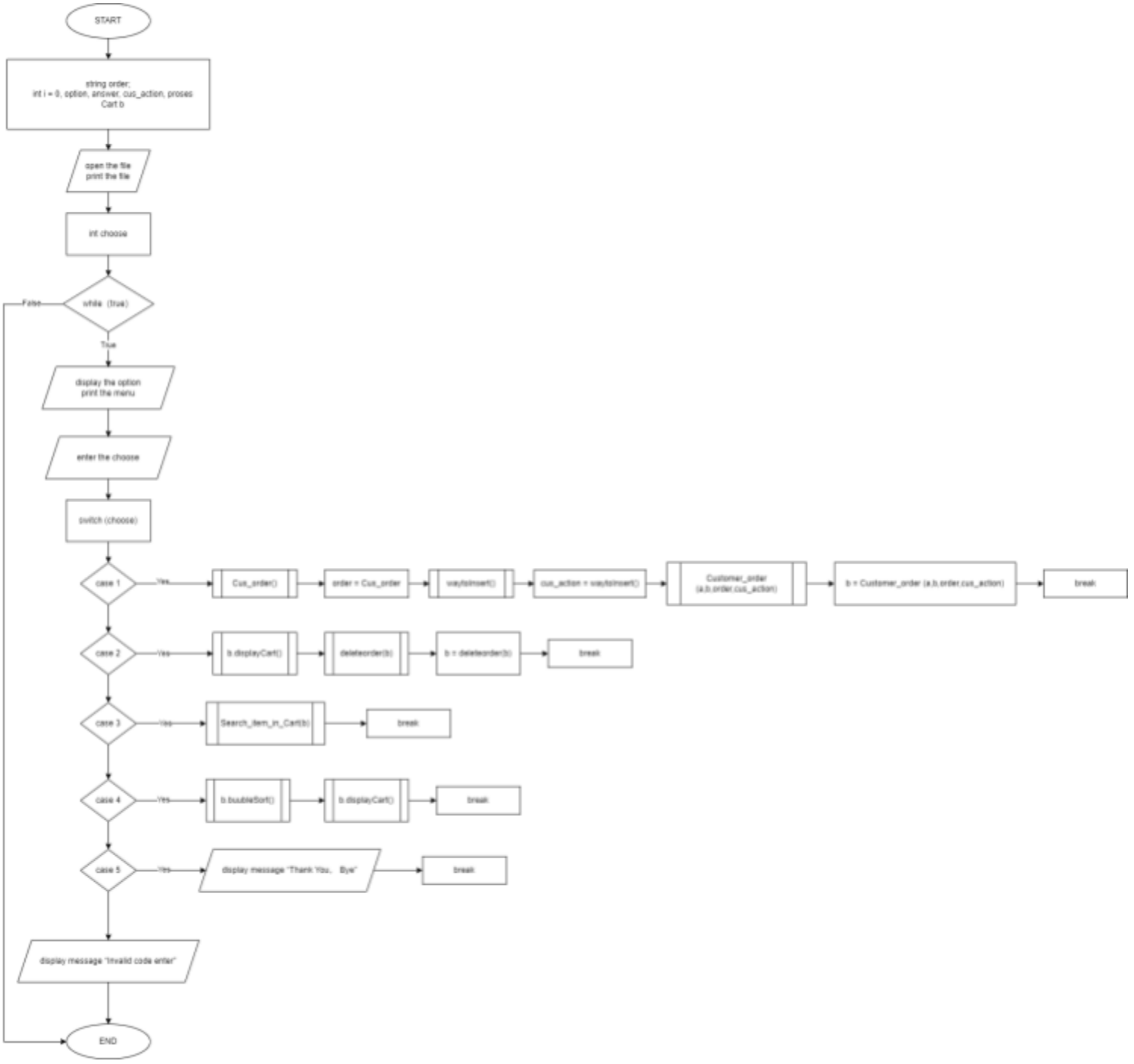
In our system, the linked list is used as a food cart that records the order from the customers from the system. The system will take down all the orders the customers have made and turn into a list inside the cart. Inside the cart, customers can add or delete the items that were selected wrongly or added more than what they need. If the customers cannot find what they ordered inside the cart as there are too many items inside, they can search the food via their code or name. The cart also can be arranged in the order that customers need such as ascending, descending or arranged based on the type. The system was implemented to delete the selected items no matter if its position is first , last or in the middle of the cart. After the order has been confirmed by the customers, the cart will show the list of the items that have been ordered before sending it to the payment system.

Class Diagram

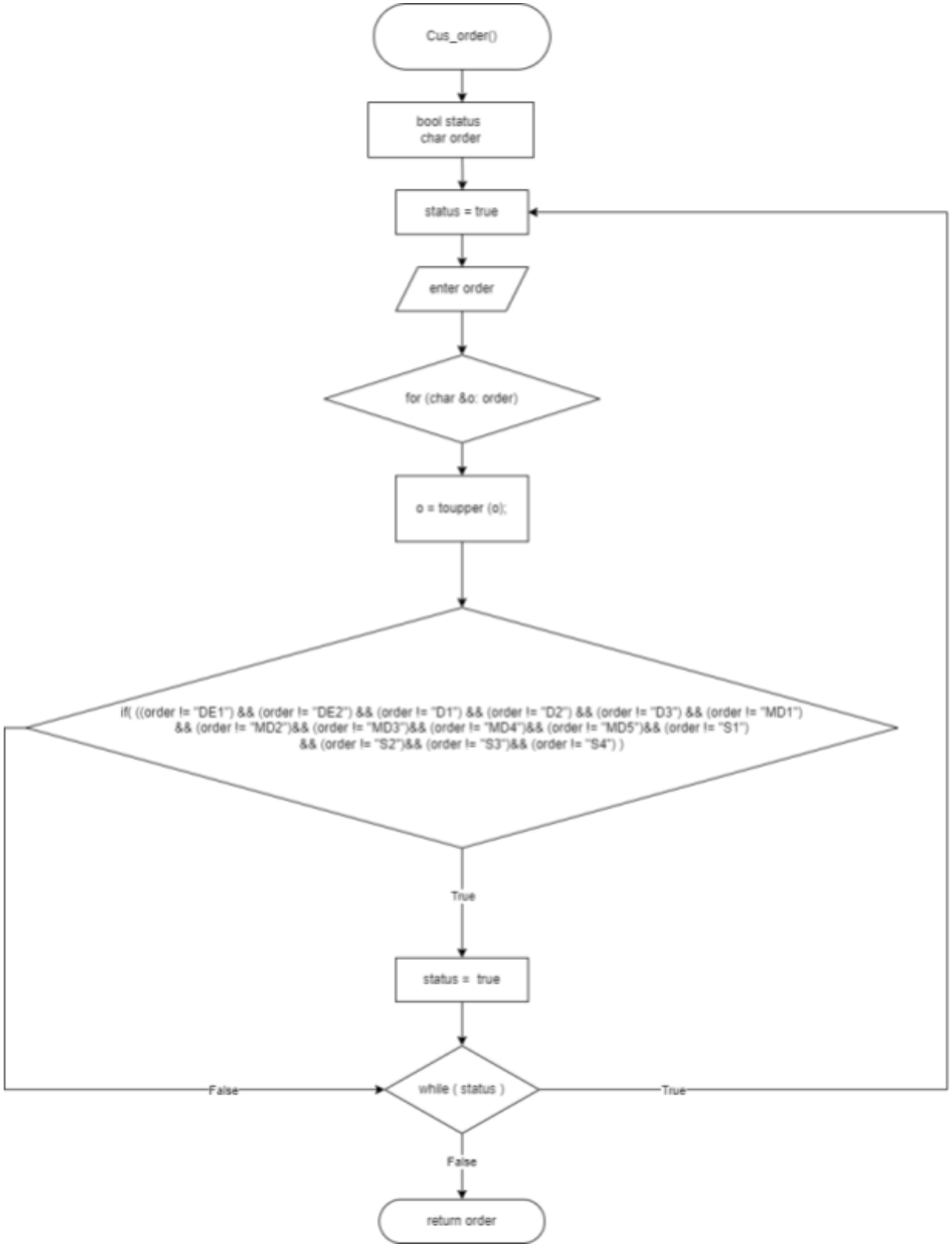


Flowchart

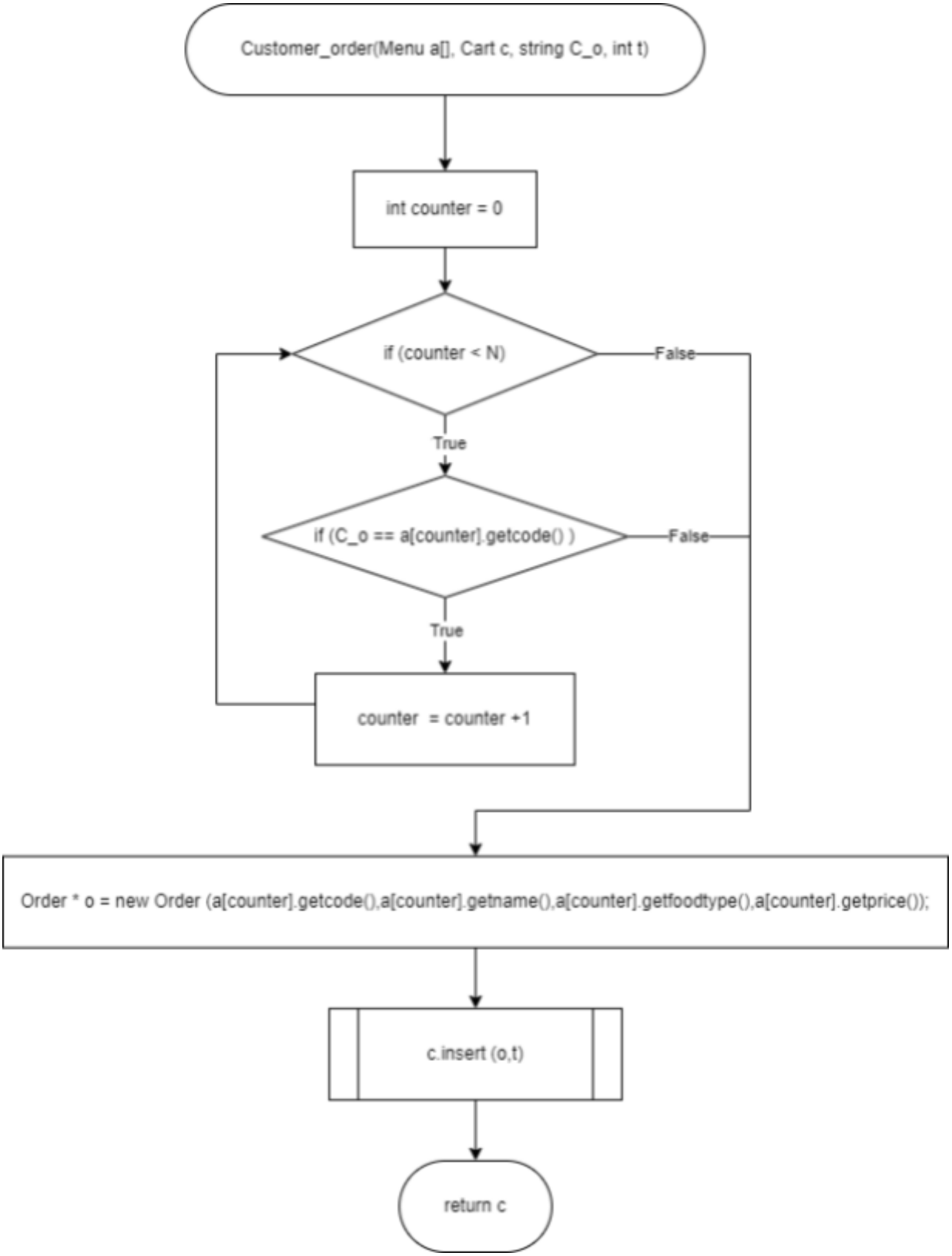
Main body



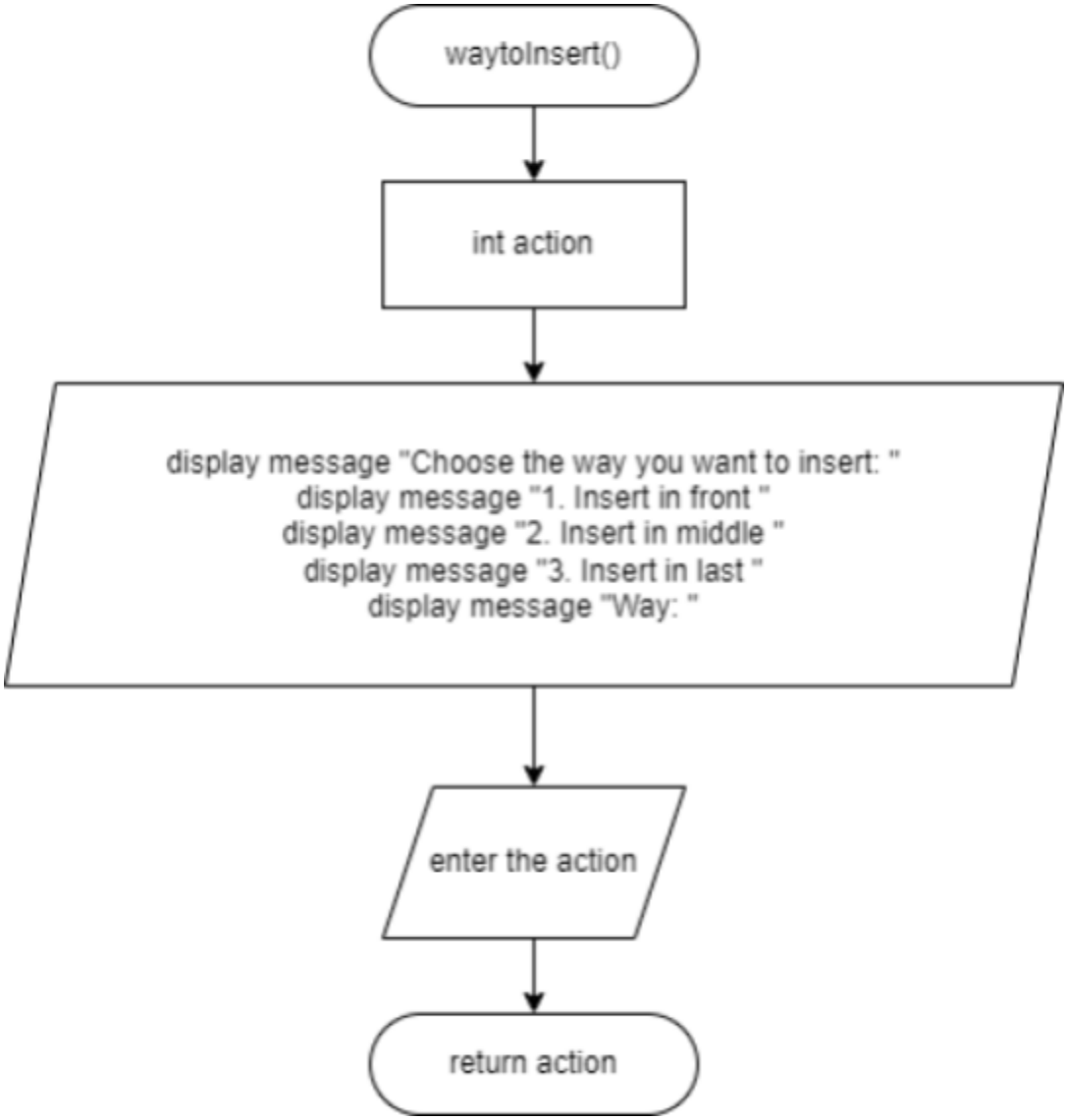
Cus_order()


Cus_order()
bool status
char order
status = true
enter order
for (char &o: order)
o = toupper (o);
if(((order != "DE1") && (order != "DE2") && (order != "D1") && (order != "D2") && (order != "D3") && (order != "MD1")
&& (order != "MD2")&& (order != "MD3")&& (order != "MD4")&& (order != "MD5")&& (order != "S1")
&& (order != "S2")&& (order != "S3")&& (order != "S4"))
True
status = true
while (status)
False
True
False
return order

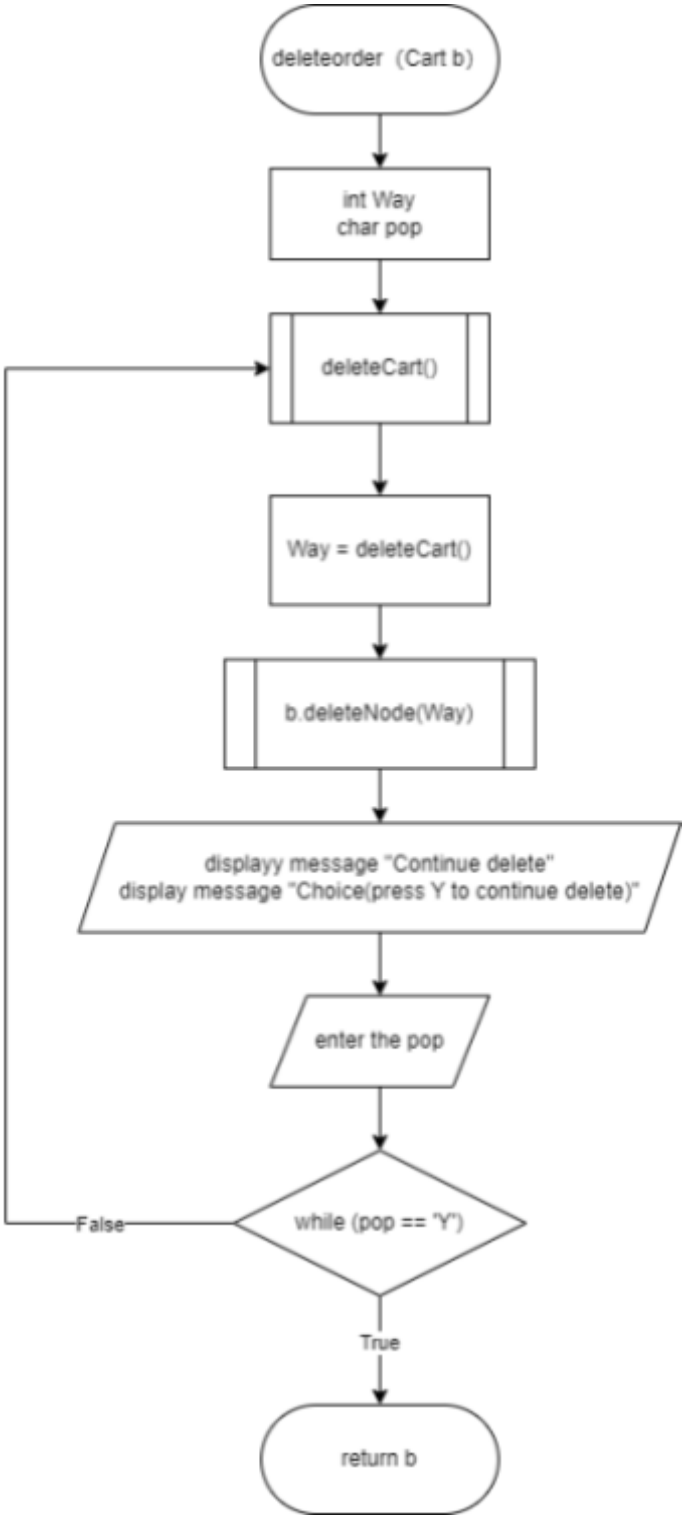
Customer_order



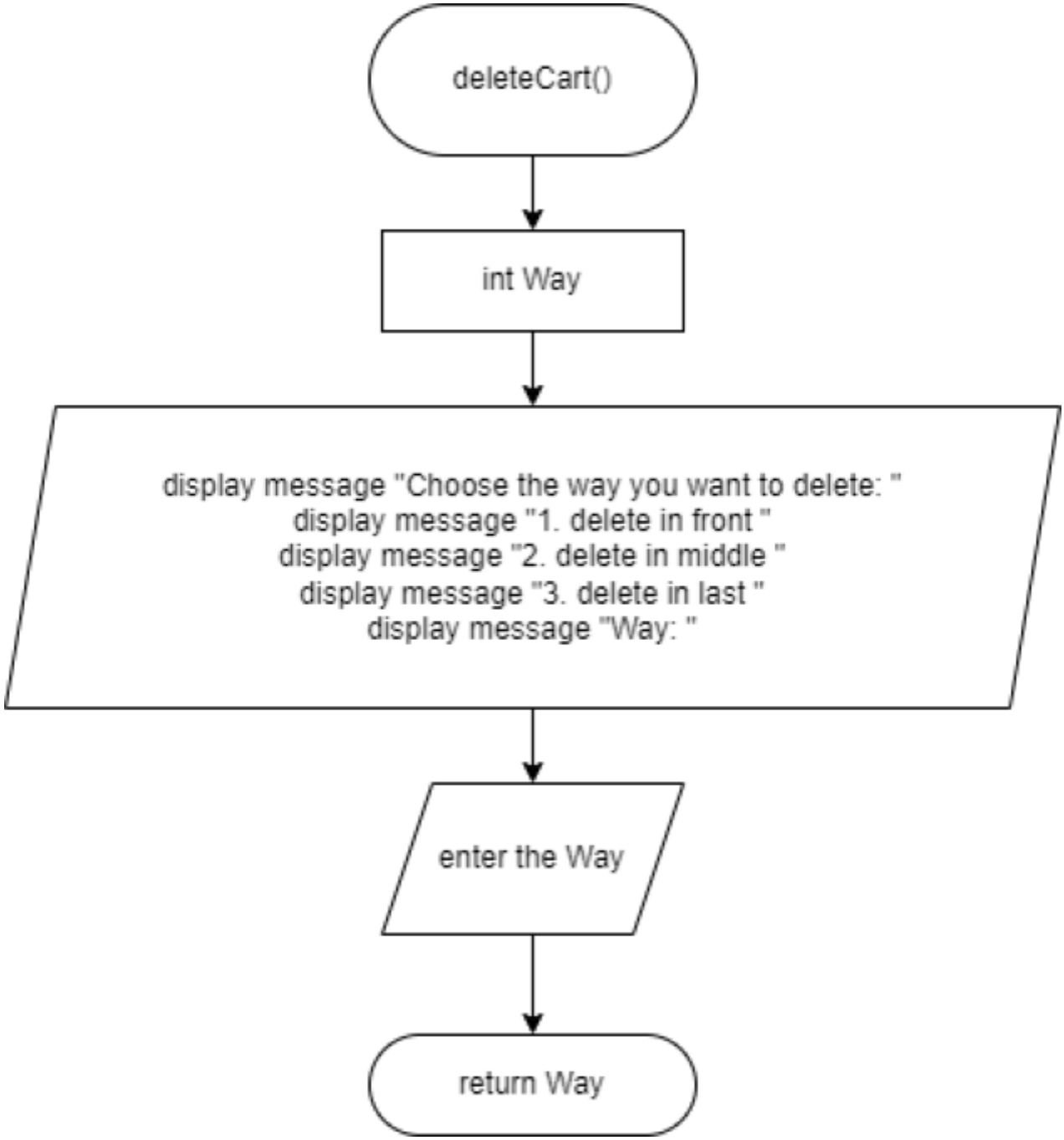
waytoInsert()



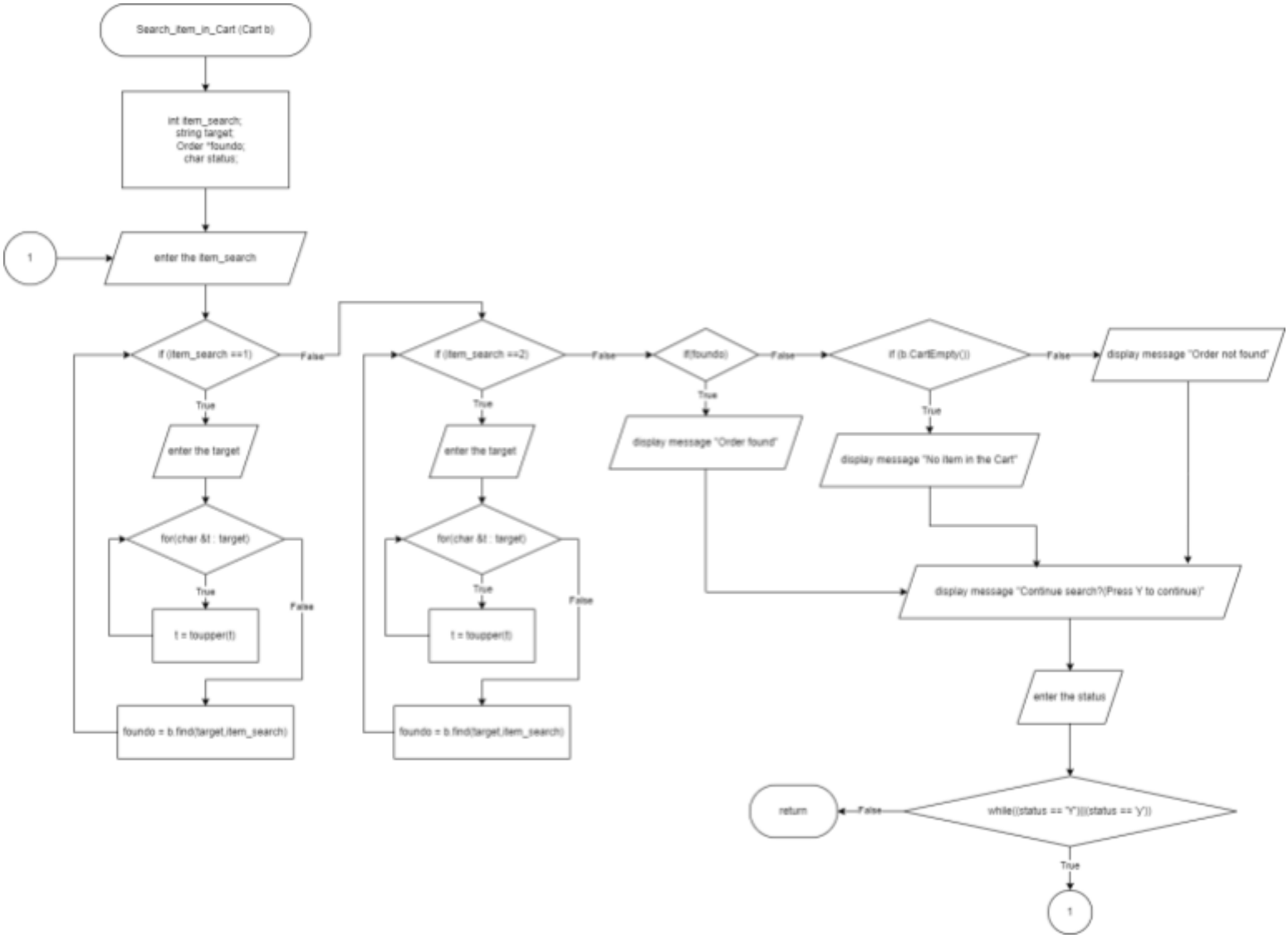
deleteorder()



deleteCart()


deleteCart()
int Way
display message "Choose the way you want to delete: "
display message "1. delete in front "
display message "2. delete in middle "
display message "3. delete in last "
display message "Way: "
enter the Way
return Way

Search_item_in_Cart()



Code

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
#define N 14
using namespace std;

class Menu{
	private:
		string code,name, foodtype;
		float price;
	public:
		Menu(string c = " ",string n = " ", string f = " ", float p =0.0 ){
			code = c;
			name = n;
			foodtype = f;
			price = p;
		}
		string getcode(){
			return code;
		}
		string getname(){
			return name;
		}
		string getfoodtype(){
			return foodtype;
		}
		float getprice(){
			return price;
		}
		void setcode(string c){
			code = c;
		}
		void setname(string n){
			name = n;
		}
		void settype(string f){
			foodtype = f;
		}
		void setprice(float p){
			price = p;
		}
```

```
		void printmenu(){
		cout << code << setw(20) << name << setw(20) << foodtype << setw(20) <<
price << endl;
		}
};

class Order{
	string code,name, foodtype;
	float price;

	public:
		Order *next;
		Order(string c ,string n, string f ,float p){
			code = c;
			name = n;
			foodtype = f;
			price = p;
			next = NULL;
		}
		string getcode(){
			return code;
		}
		string getname(){
			return name;
		}
		string getfoodtype(){
			return foodtype;
		}
		float getprice(){
			return price;
		}
		void setcode(string c){
			code = c;
		}
		void setname(string n){
			name = n;
		}
		void settype(string f){
			foodtype = f;
		}
		void setprice(float p){
			price = p;
		}
};
```

```
class Cart{
	Order *head;
	public:
		Cart(){
			head = NULL;
		}

		bool CartEmpty(){
			return head == NULL;
		}

		void insert(Order *newNode,int w){
			if(w == 1){
				if(CartEmpty()){
					head = newNode;
				}
				else{
					newNode ->next = head;
					head = newNode;
				}
			}//insert the number in front
			else if (w == 2){
				if(CartEmpty()){
					head = newNode;
				}
				else{
					int position;
					cout << "Enter the position that you want to key in:";
					cin >> position;
					Order *temp = head;
					for(int i =1; i<(position-1); i++){   // int count = 1;
while(temp->next!= NULL && count <loc){ temp = temp-> next; count ++
						temp = temp->next;
					}
					newNode->next = temp->next;
					temp->next = newNode;
				}
			}// insert the number at the middle which the user provide the location
			else if (w == 3){
				if(CartEmpty()){
					head = newNode;
				}
				else{
```

```
                                        Order *temp = head;
                                        while(temp->next != NULL){   // make it become null until
the last
                                                temp = temp->next;
                                        }
                                        temp-> next = newNode;
                                }
                        }// insert the number at the last
        }
        void deleteNode(int w){
                        Order *temp = head;
                        if(w == 1){
                                if(CartEmpty()){
                                        cout << "Does not have order can be delete" << endl;
                                }
                                else{
                                        if(temp->next != NULL){
                                        head = head->next;
                                        temp ->next = NULL;
                                        }
                                        delete temp;
                                }
                        }// delete the first number
                        else if (w == 2){
                                if(CartEmpty()){
                                        cout << "Does not have order can be delete" << endl;
                                }
                                else{
                                        int position;
                                        cout << "Enter the position that you want to key in:";
                                        cin >> position;
                                        Order *temp1;
                                        for(int i =1; i<position; i++){   // int count = 1;
while(temp->next!= NULL && count <loc){ temp = temp-> next; count ++
                                                temp1 = temp;
                                                temp = temp->next;
                                        }
                                        temp1->next = temp->next;
                                        temp->next = NULL;
                                        delete temp;
                                }
                        }// delete the number at the middle which the user provide the location
                        else if(w == 3){
                                if(CartEmpty()){
```

```
                cout << "Does not have order can be delete" << endl;
            }
            else{
                Order *temp1;
                while(temp->next != NULL){
                    temp1 = temp;
                    temp = temp -> next;
                }
                temp1->next = NULL;
                delete temp;
            }
        }// delete the last
    }
        void displayCart(){
            Order *temp;
            temp = head;
            if(temp == NULL){
                cout << "Cart is empty" << endl;
                return;
            }
            cout << "********************** Receipt ********************************" << endl
<< endl;
            cout << "-------------------------------------------------------------" << endl;
            cout << "Code" << setw(18) << "Name" << setw(20) << "Type" <<
setw(20) << "Price(RM)" << endl;
            cout << "-------------------------------------------------------------" << endl;
            while(temp){
                cout << temp->getcode() << setw(20) << temp->getname() <<
setw(20) << temp->getfoodtype() << setw(20) << temp->getprice() << endl;
                temp = temp -> next;
            }
             cout << endl;
        }
        Order* find(string target,int n){
            Order *current = head;
            while(current){
                if(n == 1){
                    if(current -> getcode() == target){
                        return current;
                    }
                }
                else if(n == 2){
                    if(current -> getname() == target){
                        return current;
```

```
                                    }
                                }
                                current = current ->next;
                        }
                        return NULL;
                }
void bubbleSort(){
    int swapped,way;
    Order *ptr1;
    Order *lptr = NULL;

    cout << "Select Receipt display type: " << endl;
        cout << "1. Arrange the Receipt by following the code (ascending order)" << endl;
        cout << "2. Arrange the Receipt by following the code (descending order)" << endl;
        cout << "3. Arrange the Receipt by following the name (ascending order)" << endl;
        cout << "4. Arrange the Receipt by following the name (descending order)" << endl;
        cout << "5. Arrange the Receipt by following the foodtype (ascending order)" << endl;
        cout << "6. Arrange the Receipt by following the foodtype (descending order)" << endl;
        cout << "Your choice:";
        cin >> way;
    // Checking for empty list
    if (head == NULL)
        return;
    if(way == 1){
    do {
        swapped = 0;
        ptr1 = head;

        while (ptr1->next != lptr){
            if (ptr1->getcode() > ptr1->next->getcode()) {
                swap(ptr1, ptr1->next);
                swapped = 1;
            }
        ptr1 = ptr1->next;
    }
        lptr = ptr1;
    } while (swapped);
                        }
    else if(way == 2){
        do {
            swapped = 0;
            ptr1 = head;

            while (ptr1->next != lptr) {
```

```
            if (ptr1->getcode() < ptr1->next->getcode()) {
               swap(ptr1, ptr1->next);
               swapped = 1;
            }
            ptr1 = ptr1->next;
         }
         lptr = ptr1;
      }while (swapped);
   }
                     else if(way == 3){
      do {
         swapped = 0;
         ptr1 = head;

         while (ptr1->next != lptr) {
            if (ptr1->getname() < ptr1->next->getname()) {
               swap(ptr1, ptr1->next);
               swapped = 1;
            }
            ptr1 = ptr1->next;
         }
         lptr = ptr1;
      }while (swapped);
   }
                     else if(way == 4){
      do {
         swapped = 0;
         ptr1 = head;

         while (ptr1->next != lptr) {
            if (ptr1->getname() < ptr1->next->getname()) {
               swap(ptr1, ptr1->next);
               swapped = 1;
            }
            ptr1 = ptr1->next;
         }
         lptr = ptr1;
      }while (swapped);
   }
            else if(way == 5){
do {
   swapped = 0;
   ptr1 = head;
```

```
            while (ptr1->next != lptr) {
                if (ptr1->getfoodtype() > ptr1->next->getfoodtype()) {
                    swap(ptr1, ptr1->next);
                    swapped = 1;
                }
                ptr1 = ptr1->next;
            }
            lptr = ptr1;
        } while (swapped);
    }
        else if(way == 6){
        do {
            swapped = 0;
            ptr1 = head;

            while (ptr1->next != lptr) {
                if (ptr1->getfoodtype() < ptr1->next->getfoodtype()) {
                    swap(ptr1, ptr1->next);
                    swapped = 1;
                }
                ptr1 = ptr1->next;
            }
            lptr = ptr1;
        } while (swapped);
    }
     }
                    void swap(Order *a, Order *b) {
            string temp_code = a->getcode();
            string temp_name = a->getname();
            string temp_foodtype = a->getfoodtype();
            float temp_price = a->getprice();

            a->setcode(b->getcode());
                    a->setname(b->getname());
            a->settype(b->getfoodtype());
            a->setprice(b->getprice());

                            b->setcode(temp_code);
            b->setname(temp_name);
                    b->settype(temp_foodtype);
            b->setprice(temp_price);
                    }
};
string Cus_order(){
```

```
	string order;
	bool status;
	do{
		status = false;
		cout << "What order do you want to place?(based on the code given)" << endl;
		cout << "Order:";
		cin >> order;
		for(char &o : order){
			o = toupper(o);
		}
		if((order != "DE1") && (order != "DE2") && (order != "D1") && (order != "D2") &&
(order != "D3") && (order != "MD1") && (order != "MD2")&& (order != "MD3")&& (order !=
"MD4")&& (order != "MD5")&& (order != "S1")&& (order != "S2")&& (order != "S3")&& (order !=
"S4")){
			cout << "Invalid code for order, Please order again" << endl;
			status = true;
		}
	}while(status);
	return order;
}

int waytoInsert(){
	int action;
	cout << "Choose the way you want to insert: " << endl;
	cout << "1. Insert in front" << endl;
	cout << "2. Insert in middle" << endl;
	cout << "3. Insert in last" << endl;
	cout << "Way: ";
	cin >> action;
	return action;
}

Cart Customer_order(Menu a[],Cart c,string C_o, int t){   // C_o = customer order
		int counter;
		for(counter = 0;  counter < N; counter++){
			if(C_o == a[counter].getcode()){
				break;
			}
		}
		Order * o = new Order
(a[counter].getcode(),a[counter].getname(),a[counter].getfoodtype(),a[counter].getprice());
		c.insert(o,t);
		return c;
}
```

```
int deleteCart(){
	int Way;
	cout << "Choose the way you want to delete: " << endl;
	cout << "1. Delete in front" << endl;
	cout << "2. Delete in middle" << endl;
	cout << "3. Delete in last" << endl;
	cout << "Way: ";
	cin >> Way;
	return Way;
}

Cart deleteorder(Cart b){
	int Way;
	char pop;
		do{
			Way = deleteCart();
			b.deleteNode(Way);
			cout << "Continue delete?" << endl;
			cout << "Choice(press Y to continue delete):";
			cin >> pop;
		}while(pop == 'Y');
		return b;
}

void Search_item_in_Cart(Cart b){
	int item_search;
	string target;
	Order *foundo;
	char status;
	do{
		cout << "Seacrh the Cart by using:" << endl;
		cout << "1. Code" << endl;
		cout << "2. Name" << endl;
		cout << "Choice: ";
		cin >> item_search;
		if(item_search == 1){
			cout << "Enter the code:";
			cin >> target;
			for(char &t : target){
			t = toupper(t);
			}
			foundo = b.find(target,item_search);
		}
```

```
		else if(item_search == 2){
			cout << "Enter the name: ";
			cin >> target;
			for(char &t : target){
			t = toupper(t);
			}
			foundo = b.find(target,item_search);
		}
		if(foundo){
			cout << "Order found" << endl;
		}
		else{
			if(b.CartEmpty()){
				cout << "No item in the Cart" << endl;
			}
			else{
			cout << "Order not found" << endl;
			}
		}
		cout << "Continue search?(Press Y to continue)";
		cin >> status;
	}while((status == 'Y')||(status == 'y'));
}

int main(){
	string menu_code, menu_name, menu_type,order;
	float menu_price;
	Menu a [N];
	int i = 0,cus_action;
	Cart b;
	ifstream file("input.txt.txt");
	if(!file){
		cout << " Error opening file" << endl;
	}
	/*
	else{
		cout << "File can run" << endl; // use for testing the file
	}
	*/
	while(getline(file,menu_code,',')){
		getline(file,menu_name,',');
		getline(file,menu_type,',');
		file >> menu_price;
		file.ignore();
```

```
            a[i].setcode(menu_code);
            a[i].setname(menu_name);
            a[i].settype(menu_type);
            a[i].setprice(menu_price);
            i++;
        }
        int choose;
        while(true){
            cout << "********** Welcome to BOBOBOY Restaurant's Ordering System
**********" << endl << endl;
            cout << "-----------------------------------------------------------" << endl;
            cout << "Code" << setw(18) << "Name" << setw(20) << "Type" << setw(20) <<
"Price(RM)" << endl;
            cout << "-----------------------------------------------------------" << endl;
            for(int i = 0; i <N; i++){
                a[i].printmenu();
            }
            cout << "Process:" << endl;
            cout << "1. Add Order" << endl;
            cout <<"2. Delete Order" << endl;
            cout << "3. Find the item you ordered in Cart" << endl;
            cout << "4. Confirm Order" << endl;
            cout << "5. Exit" << endl;
            cout << "Your Choice:";
            cin >> choose;
            switch(choose){
                case 1 :        order = Cus_order();
                                cus_action = waytoInsert();
                                b = Customer_order(a,b,order,cus_action);
                                break;
                case 2 :        b.displayCart();
                                b = deleteorder(b);
                                break;
                case 3 :        Search_item_in_Cart(b);
                                break;
                case 4 :        b.bubbleSort();
                                b.displayCart();
                                system("pause");
                                break;
                case 5 :        cout << "Thank You,Bye" << endl; exit(0); break;
                default :       cout << "Invalid code enter" << endl;
            }
            system("CLS");
```

```
        }
    }
```