Addis Ababa Science and Technology University

College of Electrical and Mechanical Engineering

ADDIS ABABA
SCIENCE AND
TECHNOLOGY
UNIVERSITY
UNIVERSITY FOR INDUSTRY

Department of Software Engineering Fundamental of programming 2 group Assignment

Group 8 member name	ID
1.Atnatiwos Amare	cep0014/14
2.Habtamu Tsegaye	cep0054/14
3.Biruk Werkneh	cep0022/14

INVENTORY SYSTEM

Introduction

This book deals with inventory management system.so this show you what is inventory system and how can we implement a c++ program for it. There are a lot of program and spreadsheet that work for inventory. however, this c++ program only for education and console based. we make it in a simple way to use also.

What is Inventory?

Inventory is one of the biggest and most important assets company has. It refers to goods, items, merchandise, and component parts to be sold as well as raw materials uses in production to produce products or repair products in order to make a profit.

inventory is also an accounting term that refers to counting or listing items of all stock in various production stages. By having precise numbers of items in inventory manufacturers and retailers can ensure they have enough stock to keep production running to build items or continue to sell items to generate the biggest profit.

Example: In a clothing manufactory the inventory would include things like different types of fabric, buttons, threads, etc. for raw materials. For a work-in-progress inventory, this could be shirts without buttons, sleeves, or collars, and for completed goods inventory this would be the products such as jeans, shirts, pants, etc. ready to be shipped.

What is the difference between stock and inventory?

In general, "stock" and "inventory" are related concepts but have slightly different meanings depending on the context. Here's how they are typically understood:

Stock:

- Stock refers to the supply or quantity of goods or products that a company or business holds or has available for sale or distribution.

- It represents the overall quantity of items that a company has in its possession or in a particular location at a given point in time.

- Stock can refer to both raw materials or components used in manufacturing and finished products that are ready for sale.

- Stock levels are important for businesses to optimize operations, meet customer demand, avoid stockouts or overstocking, and manage supply chains effectively.

Inventory:

- Inventory refers to the detailed and itemized list or record of the stock that a company maintains. It is a comprehensive record of the products, materials, or assets a company owns or holds.

- Inventory includes information such as the type, quantity, location, and value of each item or stock-keeping unit (SKU).

- It is often organized and tracked using inventory management systems or software to enable efficient control, tracking, and management of stock.

- Inventory management involves processes such as stock counting, record-keeping, tracking sales and purchases, forecasting demand, and optimizing stock levels.

In summary, stock is the overall supply or quantity of goods that a company has, while inventory is the detailed record or list of the stock that a company maintains. Stock represents the physical goods, while inventory represents the documented information about those goods. They are interconnected, with inventory providing visibility into the stock levels and enabling effective management of the supply chain and business operations.

Inventory system

An inventory management system (or inventory system) is the process by which you track your goods throughout your entire supply chain, from purchasing to production to end sales. It governs how you approach inventory management for your business.Each company will manage stock in their own unique way, depending on the nature and size of their business.

The basis of inventor system?

However, because inventory is such an important part of a business to make a profit, thus it’s important to have an understanding of key aspects of inventory such as inventory management which helps with having an effective way to purchase, receive, store, sell and dispatch inventory and turnover of inventory which is one of the primary sources of revenue generation and subsequently earnings for the company’s shareholders.

the basics of inventory management — now to develop a process for how you track and control your stock. Here’s a guide to choosing an inventory management system that matches your company’s requirements.

What is the necessity of inventory system

An inventory system is necessary for businesses and organizations to efficiently manage their inventory and ensure smooth operations. Here are some key reasons why an inventory system is essential:

1. Efficient inventory management: An inventory system allows businesses to track and manage their inventory accurately. It provides a centralized database to record and monitor stock levels, item information, and other relevant data. This helps optimize inventory levels, minimize stockouts, avoid overstocking, and improve overall inventory control.

2. Cost control and financial management: An effective inventory system helps businesses optimize their inventory levels, which can result in cost savings. By avoiding excessive inventory and minimizing stockouts, businesses can reduce carrying costs, prevent losses due to obsolescence, and improve cash flow management.

3. Accurate demand forecasting: An inventory system provides historical sales data and trends, enabling businesses to make more accurate demand forecasts. This helps in better planning, purchasing, and production decisions. By aligning inventory levels with demand, businesses can meet customer needs on time and avoid shortages or excesses.

4. Streamlined operations: A well-implemented inventory system improves workflow and efficiency by automating various inventory-related tasks. It reduces manual efforts in inventory tracking, order management, and stock replenishment processes. This enables businesses to save time, reduce errors, and enhance operational productivity.

5. Customer satisfaction: By maintaining optimal inventory levels and ensuring product availability, an inventory system helps businesses meet customer demands promptly. This improves customer satisfaction by reducing delays, avoiding backorders, and providing reliable product availability information.

6. Effective supply chain management: An inventory system facilitates effective supply chain management by providing real-time visibility into inventory levels, supplier performance, and order fulfillment. This allows businesses to monitor and control the inflow of goods, maintain supplier relationships, and optimize the entire supply chain

process.

7. Compliance and regulatory requirements: Certain industries, such as pharmaceuticals or food, have specific compliance and regulatory requirements for inventory management. An inventory system helps ensure compliance by providing accurate records, traceability, and quality control of inventory items.

Overall, an inventory system plays a critical role in optimizing inventory levels, improving operational efficiency, reducing costs, meeting customer demands, and complying with industry regulations. It enables businesses to operate more effectively, make informed decisions, and achieve better profitability and customer satisfaction.

Why you need an inventory system ?

Any venture that handles stock will need a system to accurately track and control it. Without one, you'll be working on an entirely ad-hoc basis — and you'll quickly run into situations where your business is overstocked or understocked.

Inventory systems tell you the number of components or ingredients you need to create or assemble your final product. Without this information you may end up with excess stock, eroding your bottom line, or with insufficient stock to meet customer demand.

But while you will need an inventory management system, which one you choose is entirely up to you. There are countless different systems you can adopt, ranging from simple approaches to comprehensive solutions.

Periodic vs perpetual inventory systems

There are two main ways in which companies manage inventory: periodic systems and perpetual systems. Let's take a look at the difference between the two.

Periodic inventory is a way of managing stock that relies entirely on stock taking. Businesses with a periodic system count their stock regularly — say, every 3 to 6 months — to verify stock accuracy, checking whether stock levels match up to sales figures.

Perpetual inventory is a system that involves tracking stock levels as goods are receipted, produced, sold, or returned to the store. Perpetual inventory systems tend to deliver the most up-to-date inventory figures, with less dependence on stock takes for accuracy.

Using spreadsheets to manage inventory

Plenty of businesses follow Carlos' example and start off using Excel: it can be a great way of managing inventory on a basic scale. However, this does come with a few downsides. Human errors, for instance, can easily snowball into costly problems when using spreadsheets. Plus, flexibility and accuracy can often be hard to come by.

Pros: Spreadsheets are low cost, fairly easy to set up, and work fine for companies with simple needs.

Cons: If the spreadsheet breaks, so does your inventory management system. And as your business gets more complex, it'll quickly become unwieldy.

Basic inventory management software

Many cloud-based apps come with basic inventory management functionality. While not the comprehensive tool that you'll get with dedicated inventory software, for many it's the natural next step from spreadsheets. You may find that you still need spreadsheets to cater to your specific requirements, though.

Pros: No need to find and install a new solution, most applications are cloud based

Cons: Often difficult to match to your specific needs

Dedicated inventory management software

Dedicated stock management software is developed specifically to help you track and control stock. If the software is cloud based, you'll be able to sync it up with your other cloud applications (Unleashed, for example, integrates with Xero, QuickBooks, Amazon, Shopify, Vend and hundreds of other apps), and access your data anywhere, at any time.

Pros: A powerful solution that's easy to integrate into other systems. Fully flexible, and designed specifically for the task.

Cons: Requires proper setup to function to its full potential

Enterprise resource planning systems

With enterprise resource planning (ERP), you buy a single solution to cover every aspect of business planning — instead of multiple different cloud components that integrate with each other. This typically involves choosing and installing the ‘modules’ that you need, including inventory management.

Pros: Get a single system that might cover inventory, accounting, supply chain, HR and more.

Cons: Expensive (with ongoing costs for maintenance and upgrades), requires considerable time and manpower to implement, tailored software might well do each task better.

When to upgrade your inventory management

Moving onto a dedicated inventory solution can provide massive benefits to your business, but it is a step change. Here are some signs that it might be time to upgrade.

You need more flexibility

Spreadsheets are an inflexible way to manage inventory. If you find that you need to add extra sheets to your solution, or are struggling because employees can’t access real-time data, then it might be time to upgrade to cloud-based inventory software.

You need more accuracy

Do you find it difficult to track how much of a particular item you have at any one time? Using a dedicated solution, you — and your employees — will be able to get up-to-date information on your inventory at any time, from any place.

Inventory costs are growing

Holding costs can severely undermine your bottom line when they are not kept under control. The key to minimising holding costs is only to store as much stock as you need. An efficient inventory management system helps you keep your storage efficient.

Slowing growth

Maybe your salespeople have to contact your stockroom to get find out what they have to sell, or you find yourself spending too much time manually writing reports rather than focusing on your customers. Slowing sales can be a symptom of poorly managed stock — dedicated software can help you quickly return to growth.

Problems of inventory system

The problem with the inventory system code provided is that it lacks proper error handling and validation. Here are a few specific issues:

1. No input validation: The code assumes that the user will always enter valid input, such as valid quantity and price values. But there is no validation or error handling in place to handle invalid input, leading to potential issues or crashes if the user enters unexpected values.

2. Lack of file handling safeguards: The code reads and writes to a file, but it does not handle potential errors that may occur during file operations. For example, if the file fails to open or close properly, there is no error checking and the program may proceed with invalid or corrupted data.

3. Inefficient file manipulation: The code uses a basic file system approach for adding, filling, and removing products, which involves reading and writing the entire file each time. This approach can be inefficient and slow for larger inventory lists.

4. Reliance on a specific file format: The code assumes a specific file format (CSV) for storing inventory data. While CSV is commonly used for data interchange, it has limitations and may not be suitable for handling more complex inventory data structures or relationships.

5. Minimal exception handling: The code does not properly handle exceptions or errors that may occur during program execution. For example, if a file operation fails, there is no mechanism in place to handle the error and provide appropriate feedback to the user.

To improve the inventory system, it's essential to implement proper error handling, input validation, and error reporting mechanisms. Additionally, considering the use of a more efficient database or data storage approach may be beneficial for managing inventory data effectively.

How this program usefull and what problem it come up with

First in the declaration.

Declared a class name " Inventory " that has private member data within struct object which is stock item name, Price, stock id and last date and time. All data member is declare within a class.

After the declaration.

with in the main function we have intro function that have 5 choicest add new products function 2 for fill product function 3 for remove products function 4 for available stock list function and 5 for exit function .

1. With in add new products function it request both the dealer or employees to enter a stock item, quantity and price. When the finished to add new products it save with current time. stock id Then uses to give 3 digit random number that begins with capital A for each stock items. Once the random number given to that specific item it didn't change at all afterward. Then the user finish the program save with spreadsheet file with table. Finally it send you a message "you have successfully add items to inventory " and display all that i entered with table format which means stock id, the time, item name, quantity and price respectively.

2. With fill product function it ask both the dealer or employees to enter product name if that exist before in the file it ask the user to enter the quantity and add with the previous quantity. and ask the price and overwrite the price. Finally it send you a successful message and display Available stock list function .

3. With remove products function it prompt both the employees or dealer to enter a stock item name and delete that stock item name, price and quantity completely from the file. Finally it send you a successful message and display Available stock list function.

4. With Available stock list function it display all the items with table that saved in the file.

5. With exit function it finish the program.

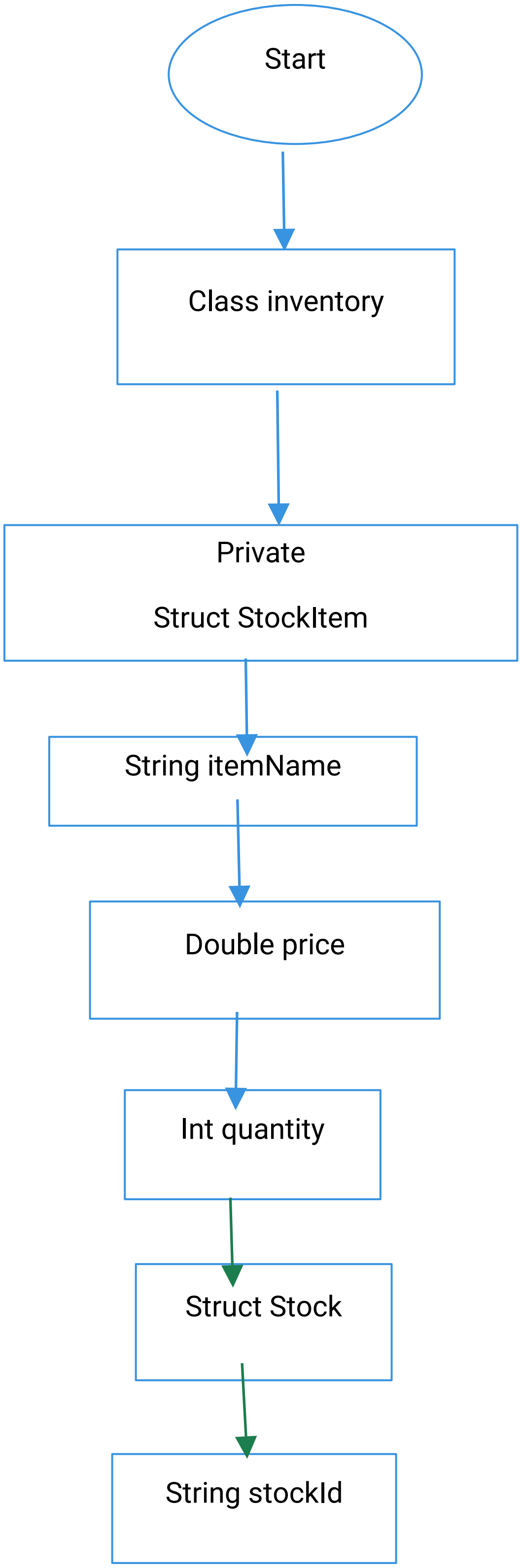
PSEUDOCODE

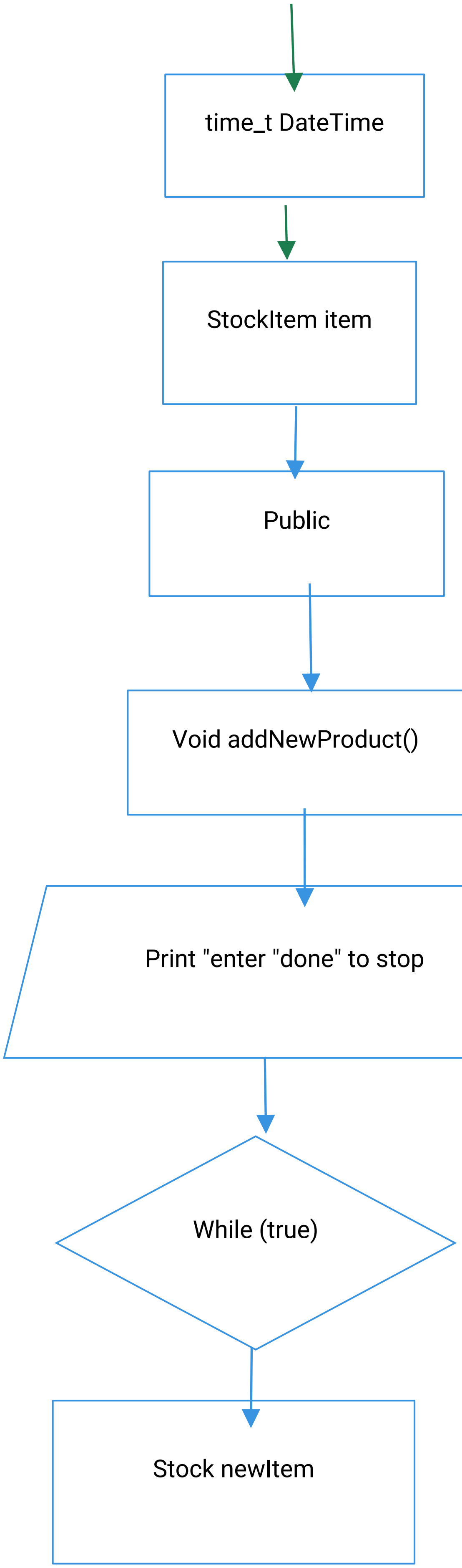
```
class Inventory {

  private:

    // Define StockItem structure

    struct StockItem {

      string itemName;

      double price;
      int quantity;
    }
    // Define Stock structure
    struct Stock {
      string stockId;
      time_t dateTime;
      StockItem item;
    }
  public:
    // Function to add a new product to the inventory
    void addNewProduct() {
      // Display instructions to the user
      // Loop until the user is done adding products
        // Prompt the user for the item name
        // Check if the user wants to stop adding products
          // Break the loop
        // Prompt the user for quantity and price
        // Set the current date and time for the Stock instance
        // Generate a stock ID
        // Save the new item to the spreadsheet
        // Display success message to the user
      // Display the updated inventory
      // Prompt the user to press any key
      // Clear the screen
      // Go back to the main menu
    }
```

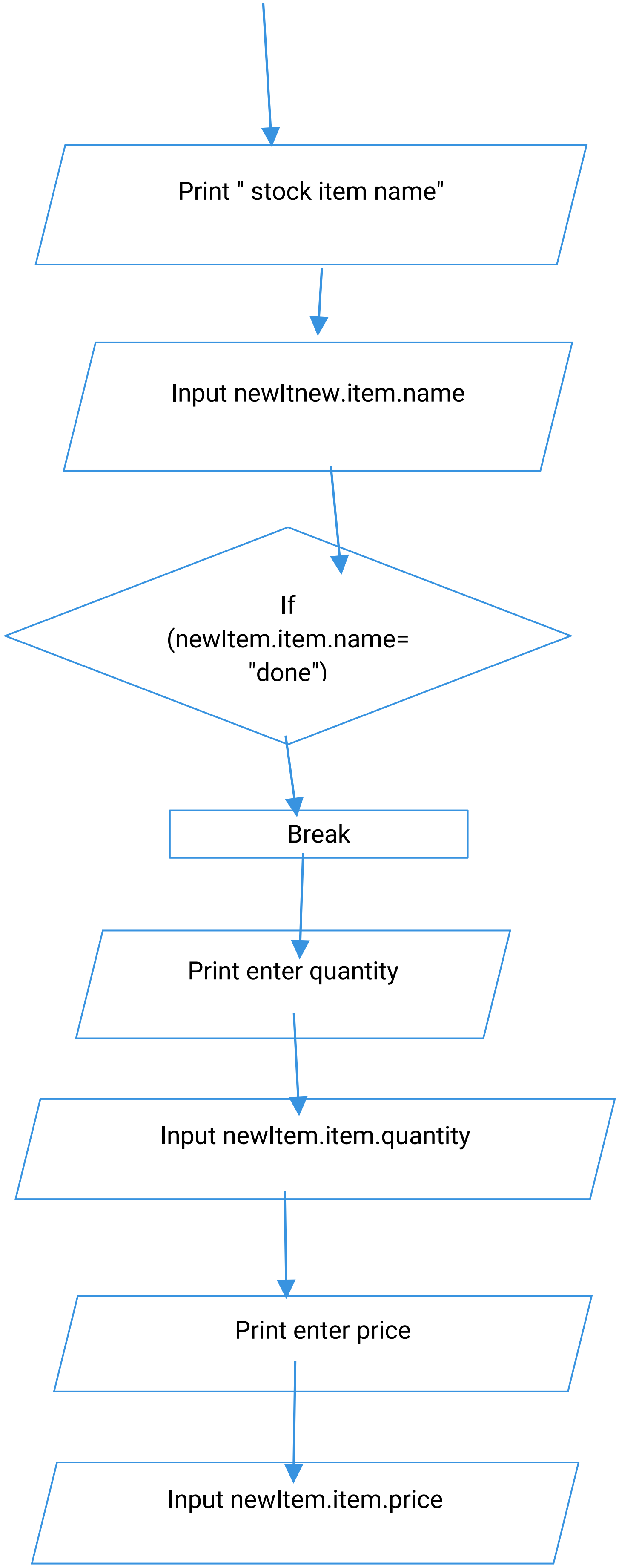
```
    // Function to fill a product in the inventory
    void fillProduct() {
      // Display the current inventory
      // Prompt the user for the product name
      // Get the Stock instance for the given product name
      // Check if the product exists in the inventory
        // Display error message and return
      // Prompt the user for the quantity to add
      // Update the quantity and price of the item
      // Update the item in the inventory
      // Display success message to the user
      // Display the updated inventory
      // Prompt the user to press any key
      // Clear the screen
      // Go back to the main menu
void removeProduct() {
  // Clear the screen
  // Display the current inventory
  // Prompt the user for the product name to remove
  // Get the Stock instance for the given product name
  // Check if the product exists in the inventory
    // Display error message and return
  // Remove the product from the inventory
  // Display success message to the user
  // Display the updated inventory
  // Prompt the user to press any key
  // Clear the screen
  // Go back to the main menu
}
void availableStockList() {
  // Open the inventory file
  // Check if the file opened successfully
    // Display error message and return
  // Display header for the stock list
  // Read each line from the file
  // Parse the line to extract stock information
  // Display the stock information
  // Close the file
}
private:
string generateStockId() {
  // Generate a random stock ID
  // Return the generated stock ID
}
void saveToSpreadsheet(const Stock &item) {
  // Open the inventory file in append mode
  // Check if the file opened successfully
    // Display error message and return
  // Write the item details to the file
  // Close the file
}
```

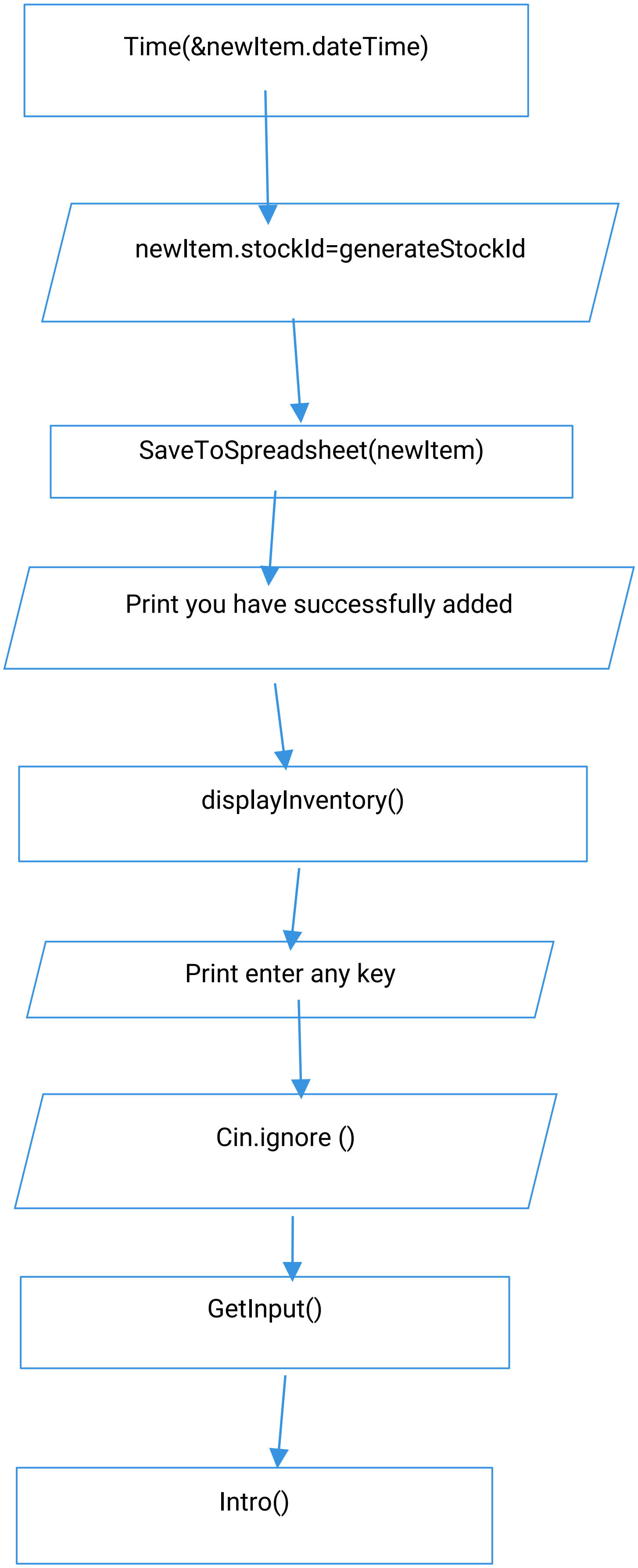
```
Stock findItem(const string &itemName) {
  // Open the inventory file
  // Initialize an empty Stock instance
  // Read each line from the file
  // Parse the line to extract stock information
  // Check if the item name matches the given item name
    // Set the Stock instance with the matched details
    // Break the loop
  // Close the file
  // Return the Stock instance
void updateItem(const Stock &item) {
  // Open the inventory file for reading
  // Open a temporary file for writing
  // Initialize variables for stock details
  // Read each line from the file
  // Parse the line to extract stock information
  // Check if the stock ID matches the given item's stock ID
    // Write the updated item details to the temporary file
  // Close both files
  // Remove the original inventory file
  // Rename the temporary file to "inventory.csv"
}
void removeItem(const Stock &item) {
  // Open the inventory file for reading
  // Open a temporary file for writing
  // Initialize variables for stock details
  // Read each line from the file
  // Parse the line to extract stock information
  // Check if the stock ID does not match the given item's stock ID
    // Write the stock details to the temporary file
  // Close both files
  // Remove the original inventory file
  // Rename the temporary file to "inventory.csv"
}
time_t parseDateTime(const string &dateTimeStr) {
  // Parse the date and time string into a tm struct
  // Convert the tm struct to a time_t value using mktime
  // Return the time_t value
}
void displayInventory() {
  // Call the availableStockList() function
}
int main() {
  // Display welcome message
    // Wait for user input
    // Call intro() to display menu and handle user's choice
    return 0;
}void intro() {
  // Clear the screen
  // Display menu options
  // Prompt the user for choice
  // Take user input for choice
// Based on the choice, call appropriate functions or exit the program.
```

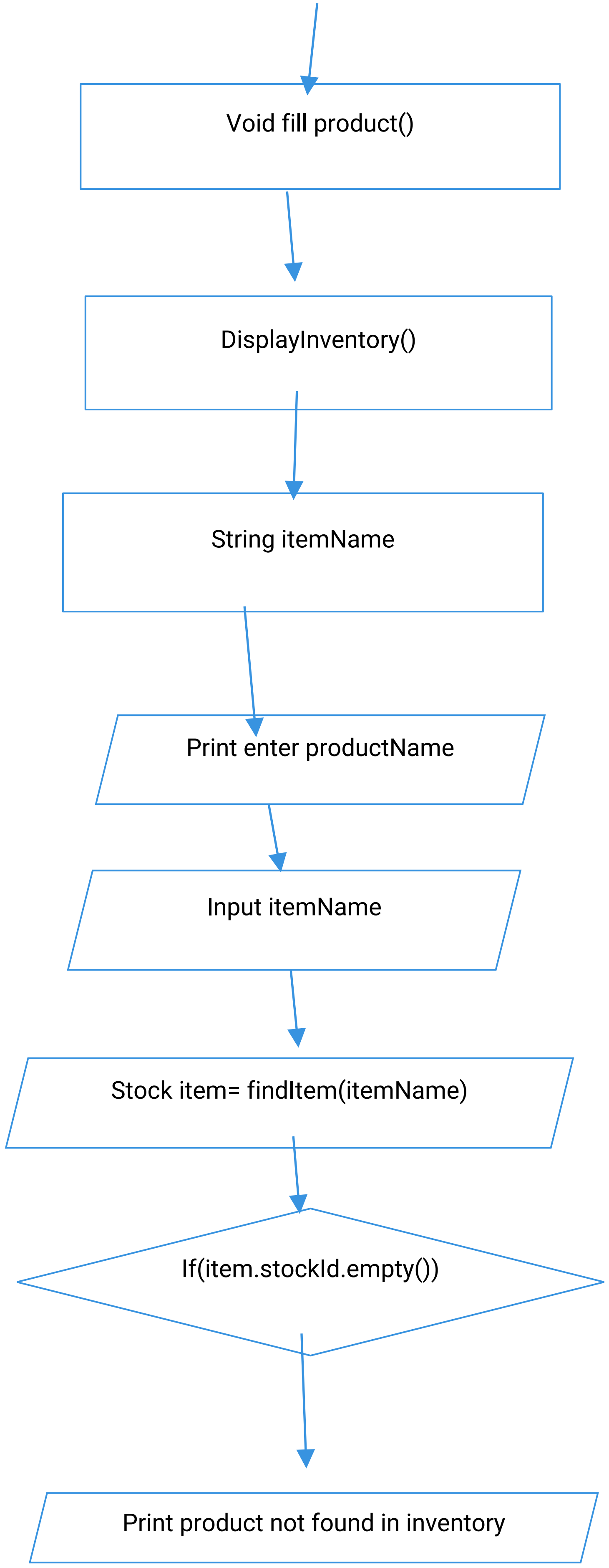
FLOWCHART

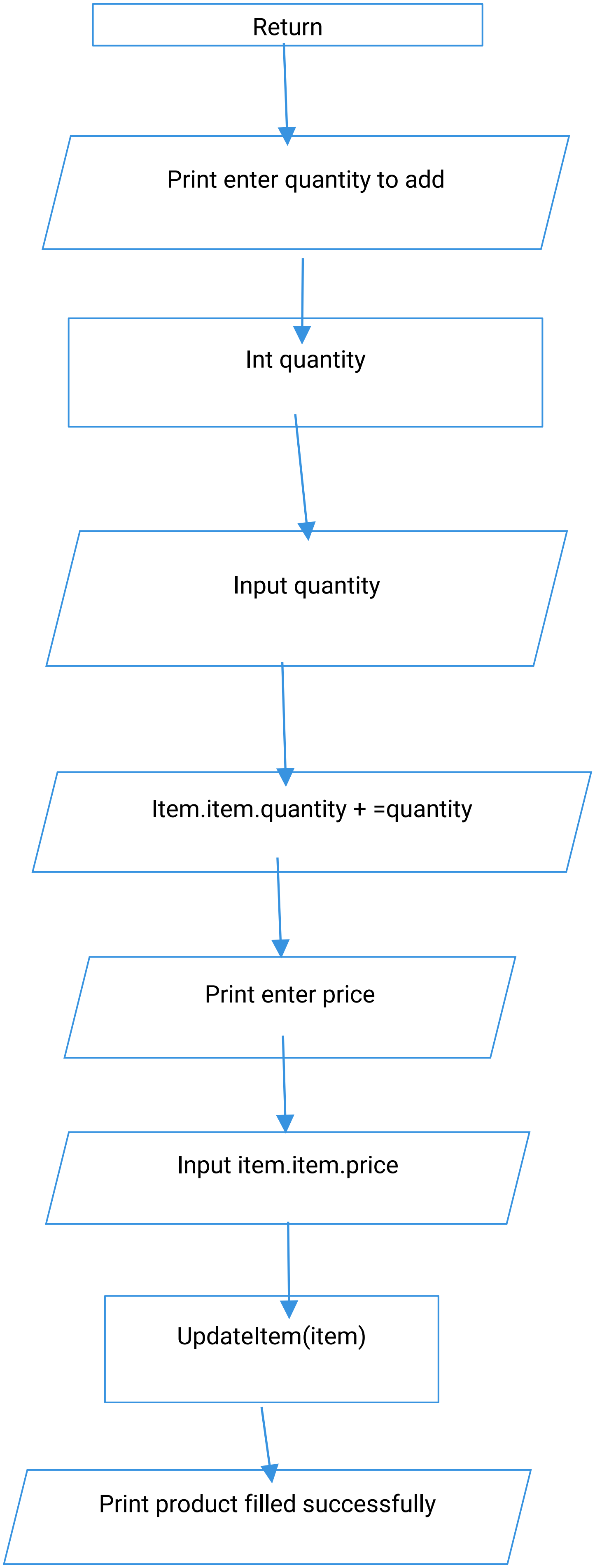


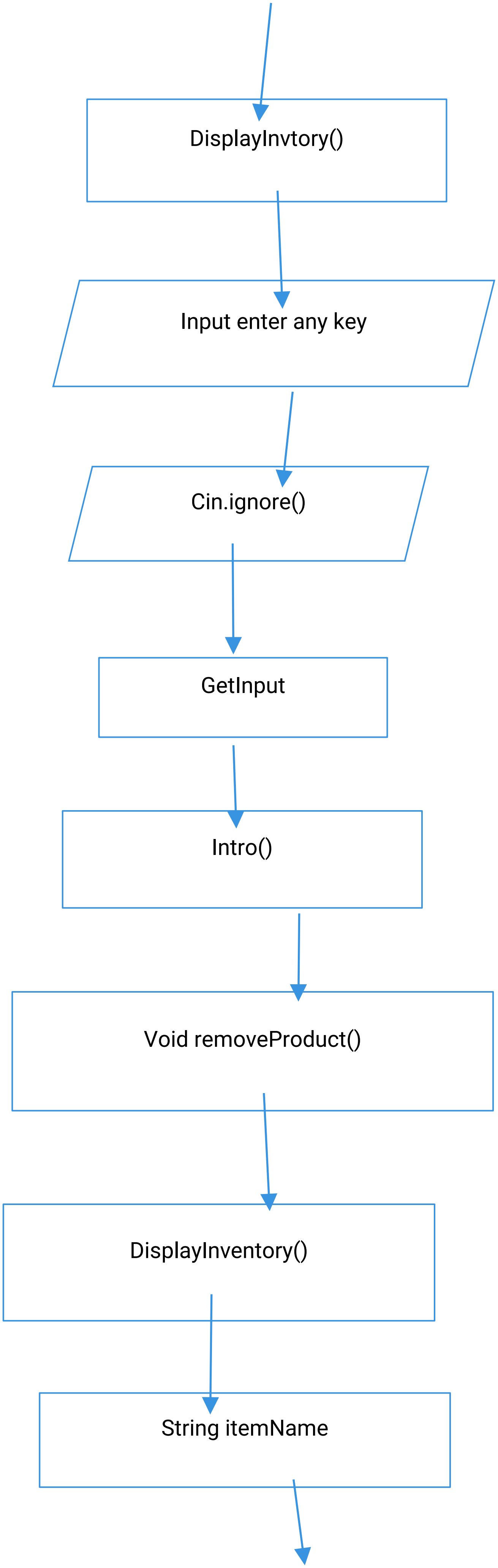

time_t DateTime
StockItem item
Public
Void addNewProduct()
Print "enter "done" to stop
While (true)
Stock newItem

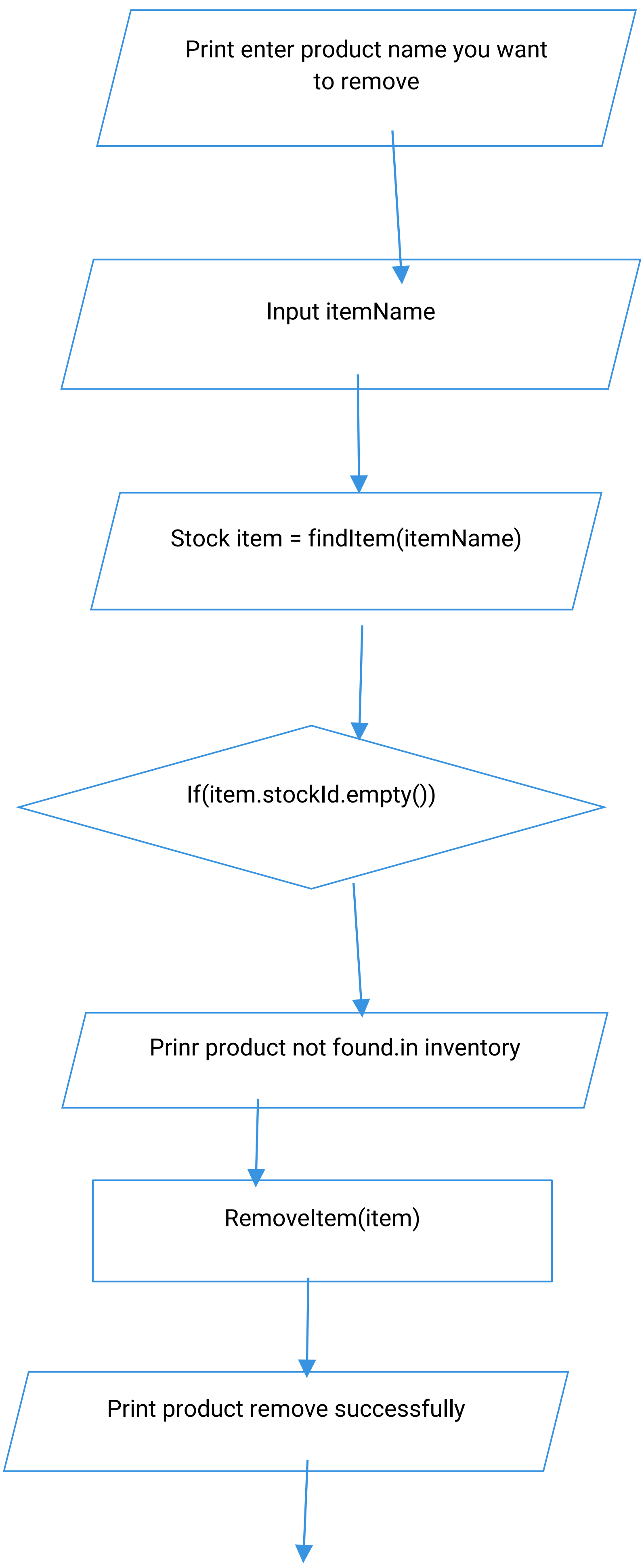

Print " stock item name"
Input newItnew.item.name
If
(newItem.item.name=
"done")
Break
Print enter quantity
Input newItem.item.quantity
Print enter price
Input newItem.item.price

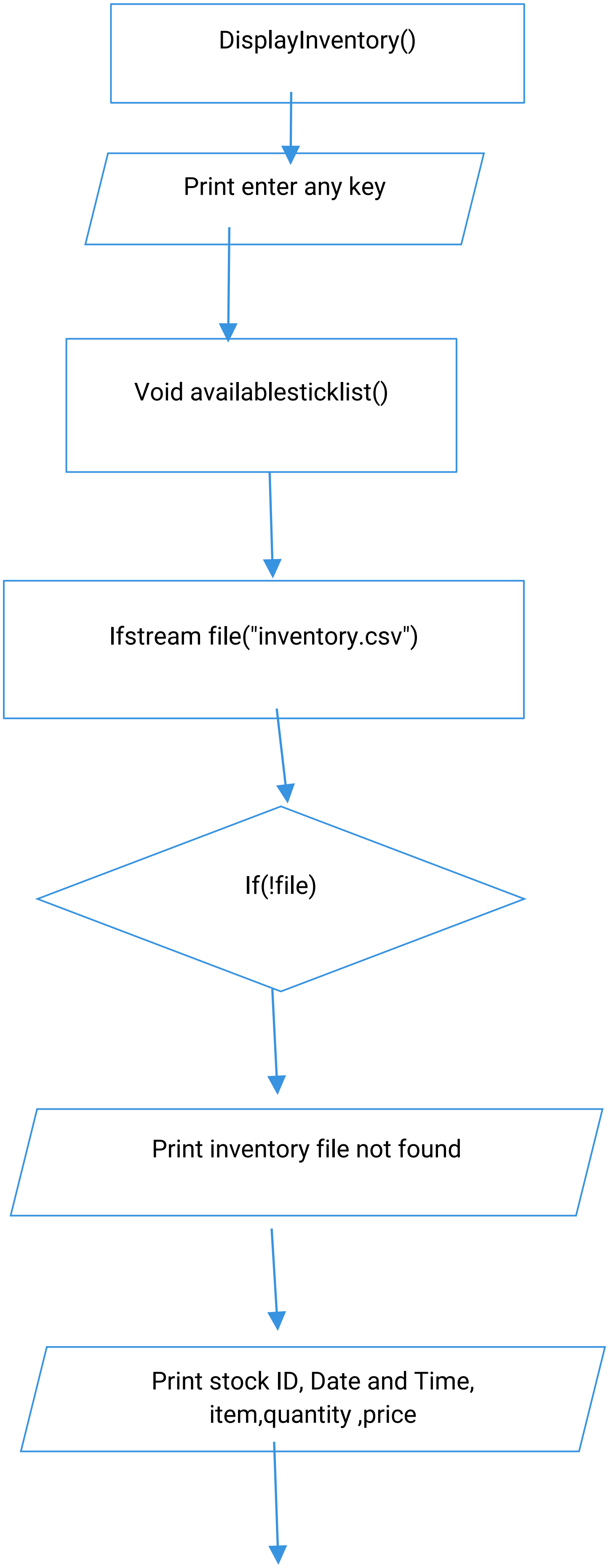

Time(&newItem.dateTime)
newItem.stockId=generateStockId
SaveToSpreadsheet(newItem)
Print you have successfully added
displayInventory()
Print enter any key
Cin.ignore ()
GetInput()
Intro()

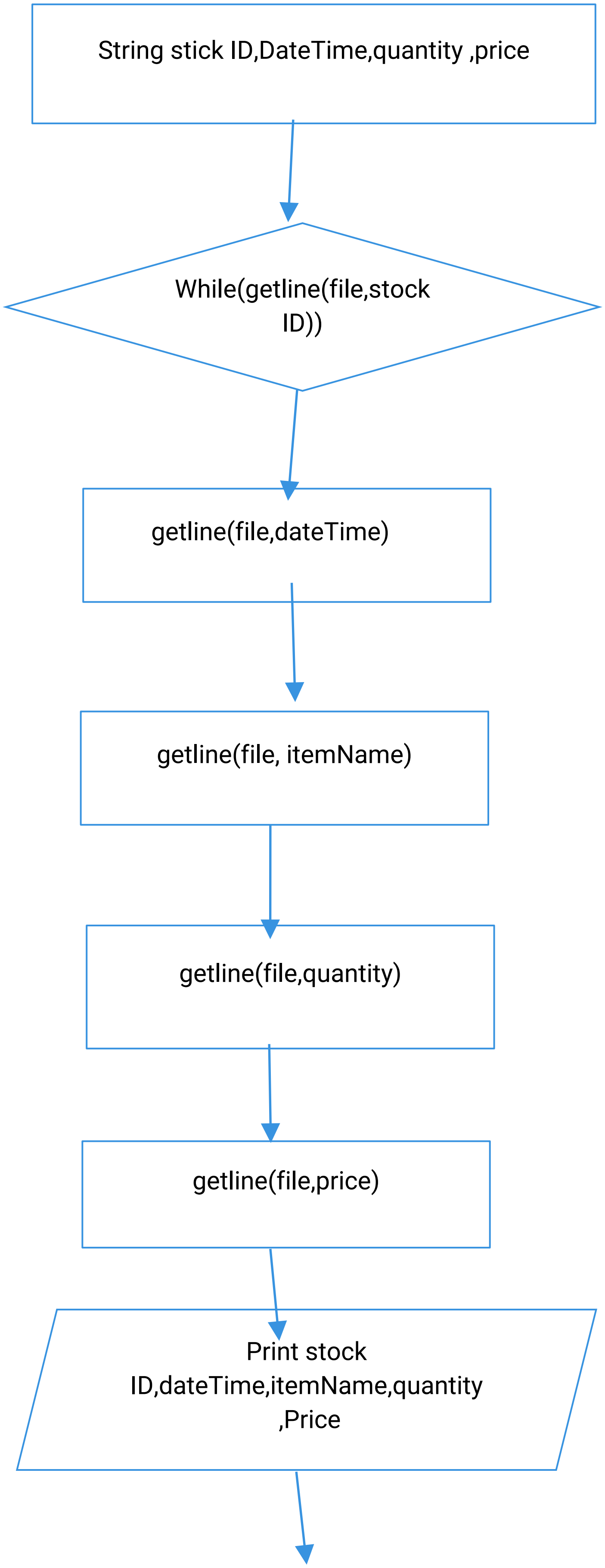

Void fill product()
DisplayInventory()
String itemName
Print enter productName
Input itemName
Stock item= findItem(itemName)
If(item.stockId.empty())
Print product not found in inventory

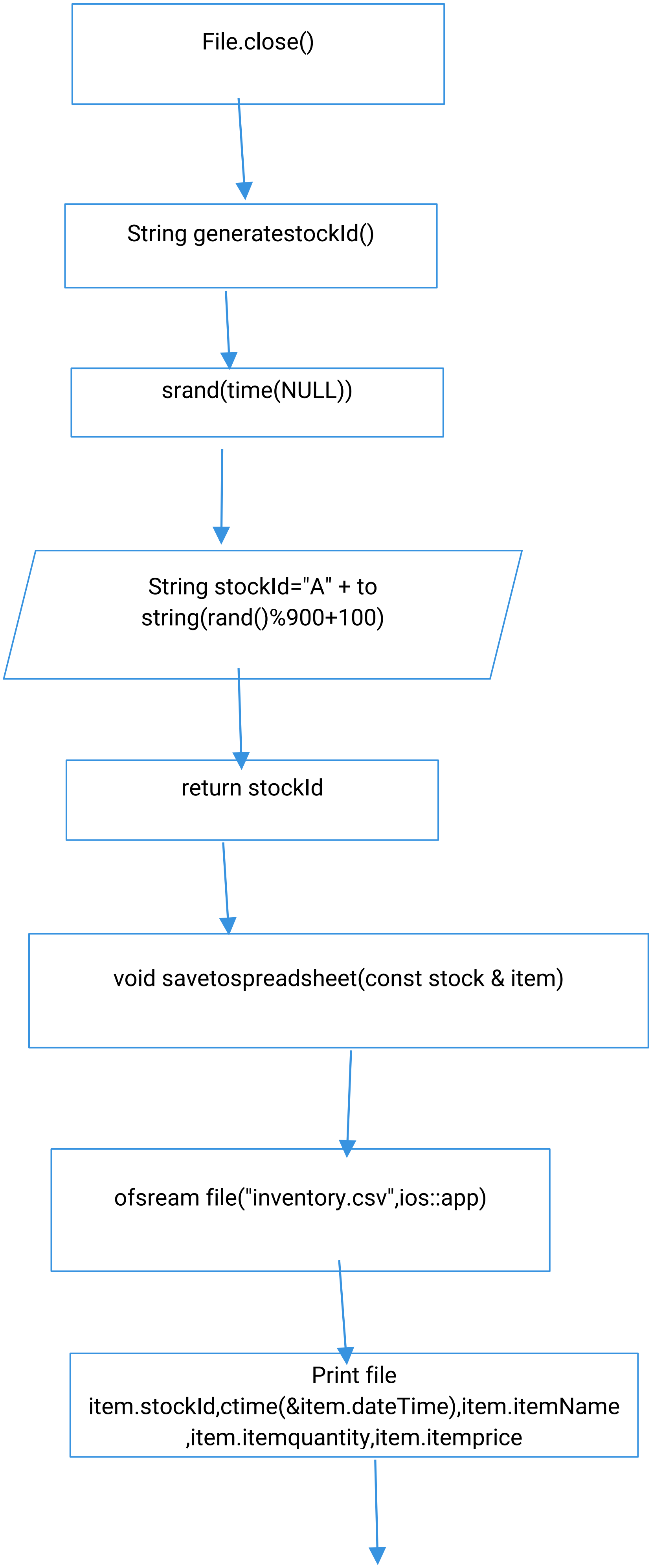

Return
Print enter quantity to add
Int quantity
Input quantity
Item.item.quantity + =quantity
Print enter price
Input item.item.price
UpdateItem(item)
Print product filled successfully

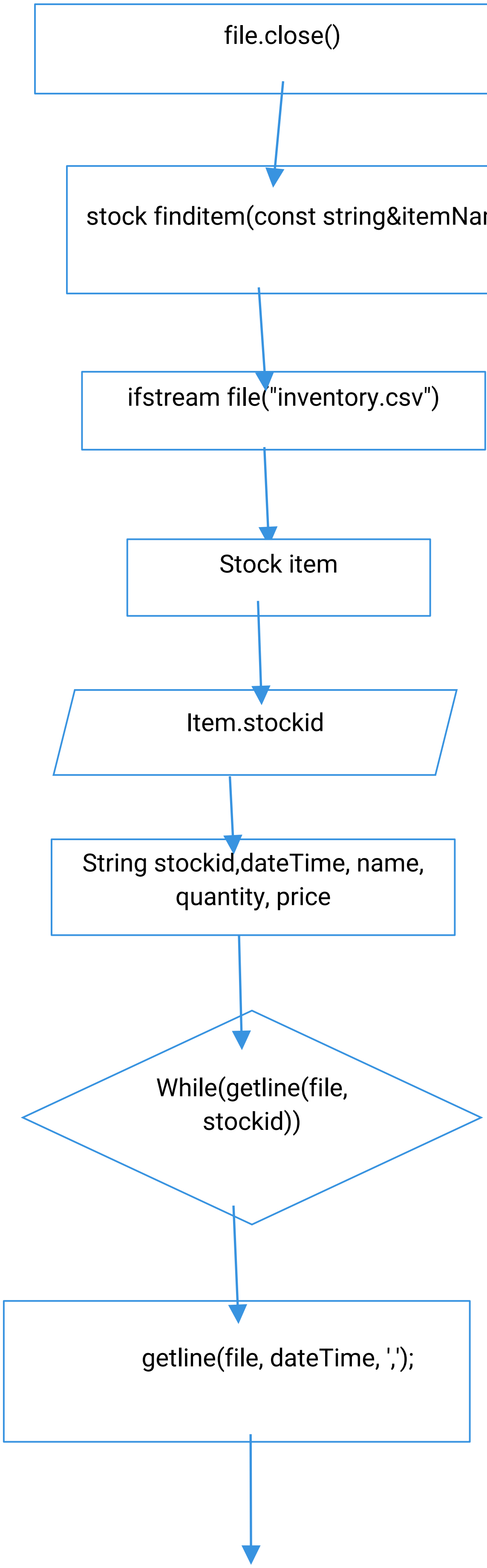

DisplayInvtory()
Input enter any key
Cin.ignore()
GetInput
Intro()
Void removeProduct()
DisplayInventory()
String itemName

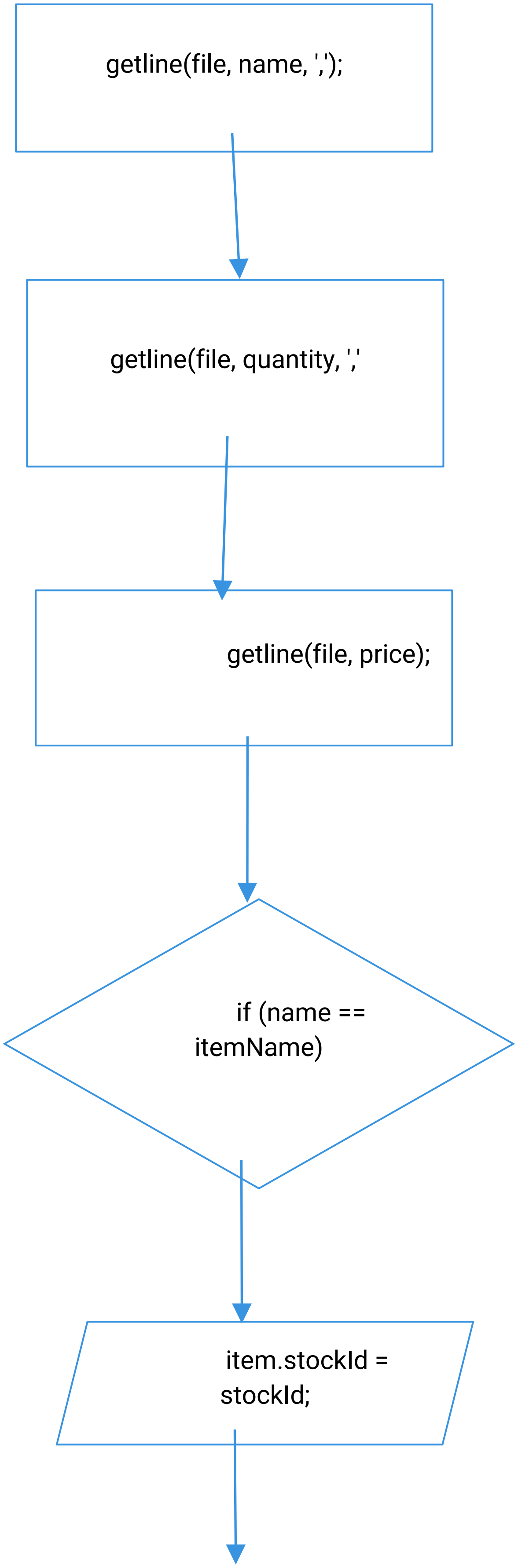

Print enter product name you want to remove
Input itemName
Stock item = findItem(itemName)
If(item.stockId.empty())
Prinr product not found.in inventory
RemoveItem(item)
Print product remove successfully

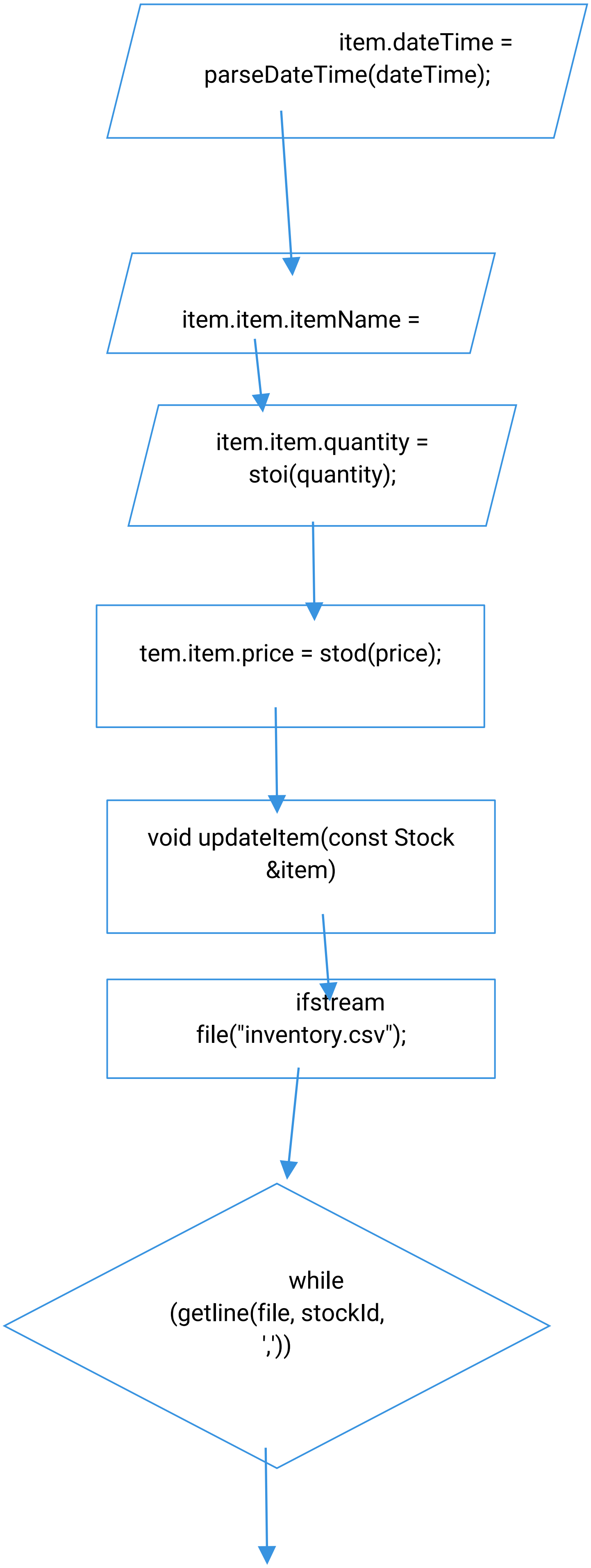

DisplayInventory()
Print enter any key
Void availablesticklist()
Ifstream file("inventory.csv")
If(!file)
Print inventory file not found
Print stock ID, Date and Time,
item,quantity ,price

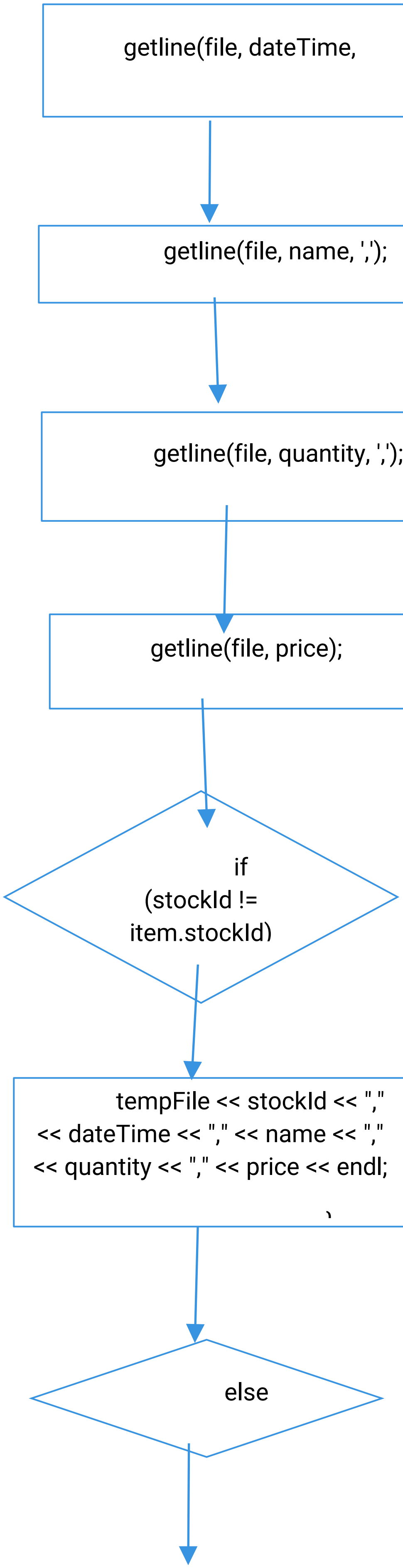

String stick ID,DateTime,quantity ,price
While(getline(file,stock ID))
getline(file,dateTime)
getline(file, itemName)
getline(file,quantity)
getline(file,price)
Print stock ID,dateTime,itemName,quantity ,Price

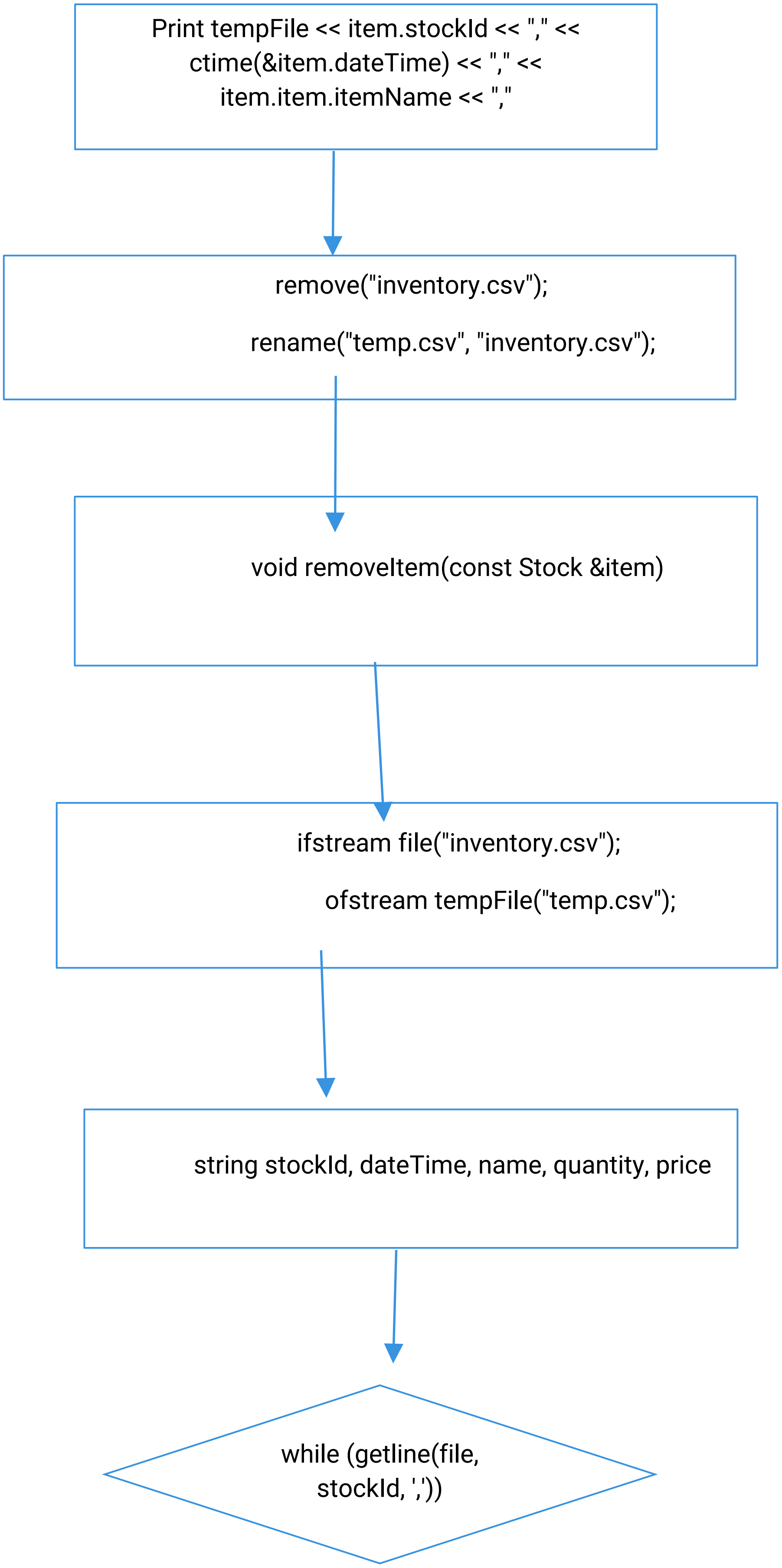

File.close()
String generatestockId()
srand(time(NULL))
String stockId="A" + to string(rand()%900+100)
return stockId
void savetospreadsheet(const stock & item)
ofsream file("inventory.csv",ios::app)
Print file item.stockId,ctime(&item.dateTime),item.itemName ,item.itemquantity,item.itemprice


file.close()
stock finditem(const string&itemName)
ifstream file("inventory.csv")
Stock item
Item.stockid
String stockid,dateTime, name, quantity, price
While(getline(file, stockid))
getline(file, dateTime, ',');

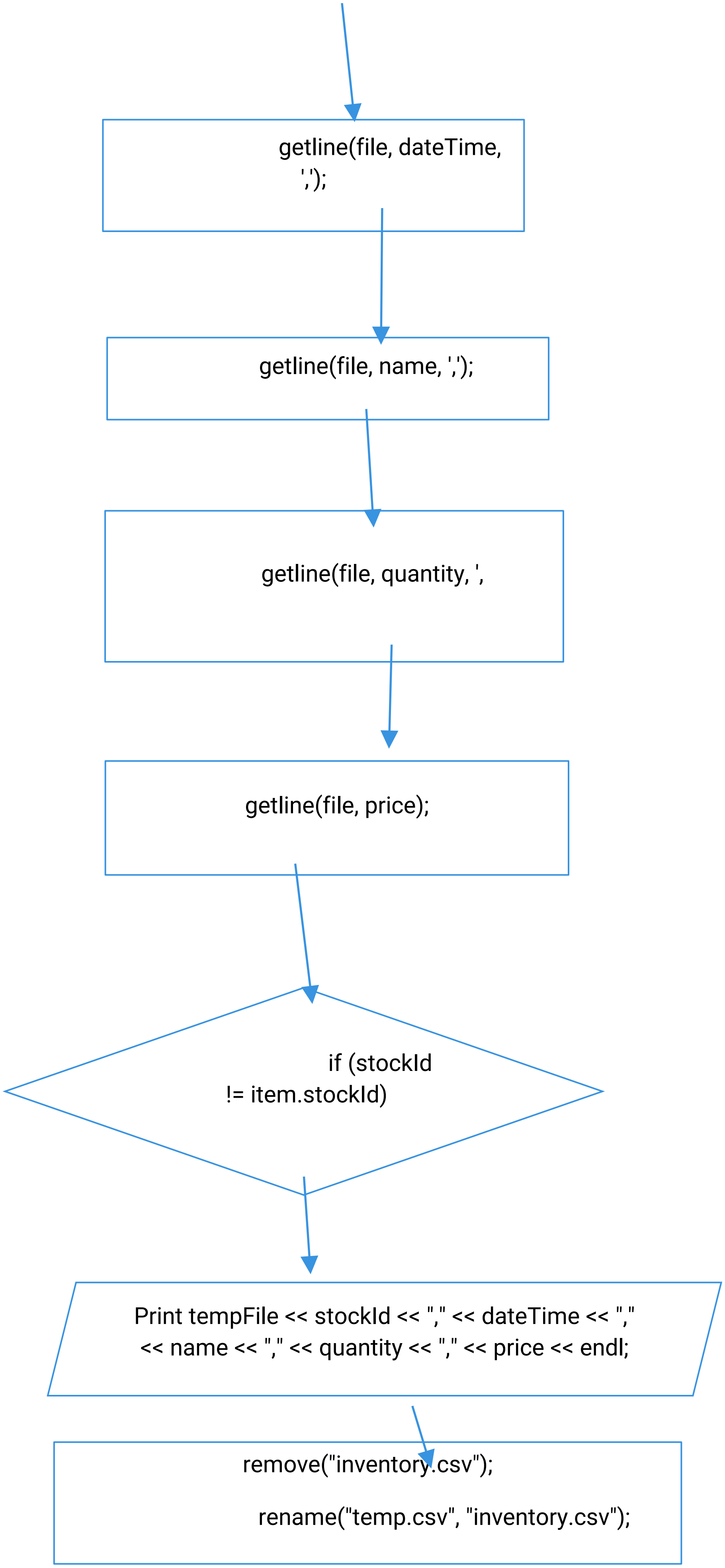

getline(file, name, ',');
getline(file, quantity, ','
getline(file, price);
if (name ==
itemName)
item.stockId =
stockId;

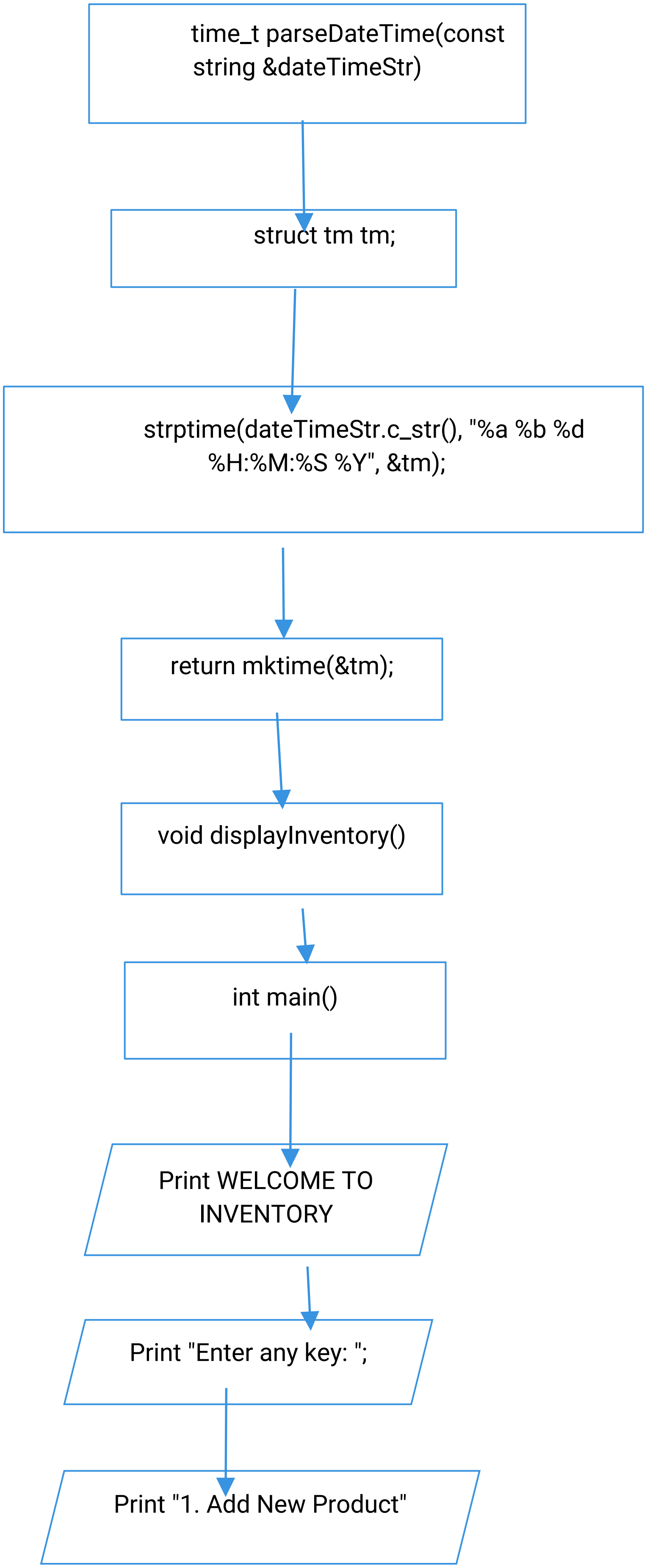

item.dateTime = parseDateTime(dateTime);
item.item.itemName =
item.item.quantity = stoi(quantity);
tem.item.price = stod(price);
void updateItem(const Stock &item)
ifstream file("inventory.csv");
while (getline(file, stockId, ','))

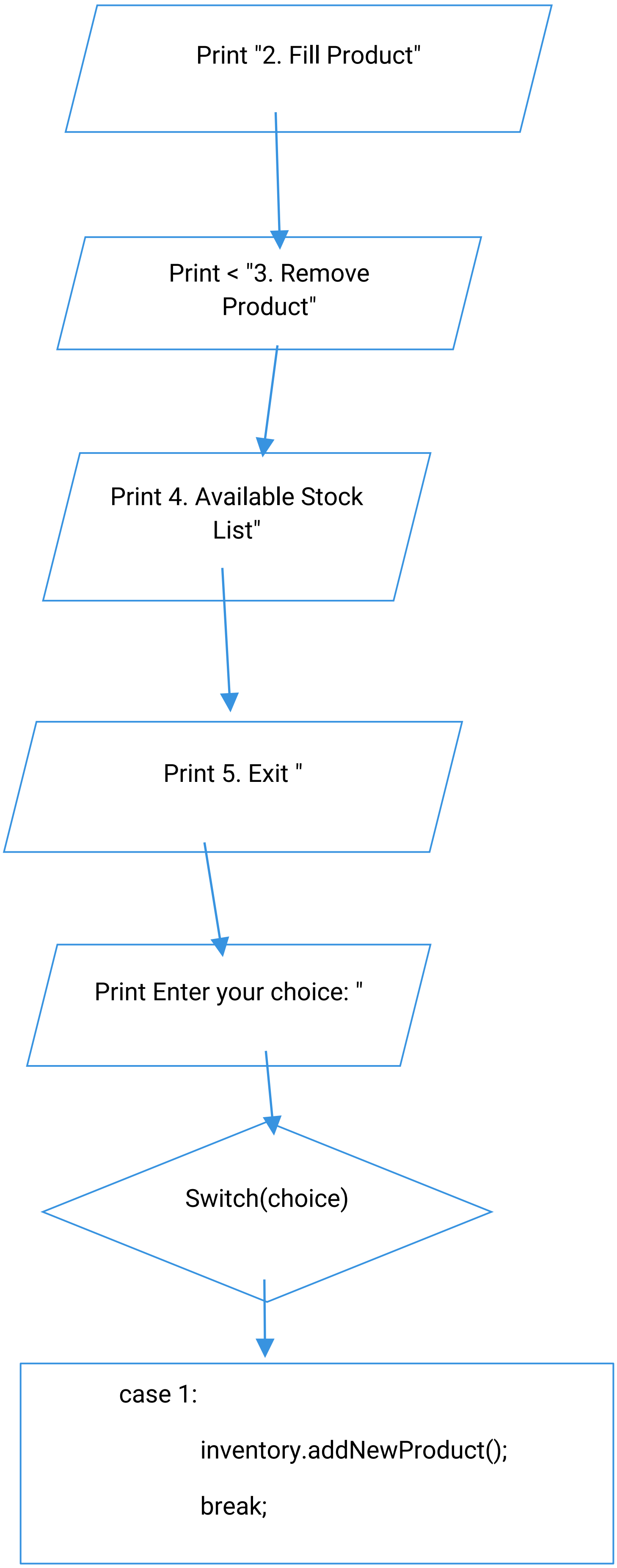

getline(file, dateTime,
getline(file, name, ',');
getline(file, quantity, ',');
getline(file, price);
if
(stockId !=
item.stockId)
tempFile << stockId << ","
<< dateTime << "," << name << ","
<< quantity << "," << price << endl;
else



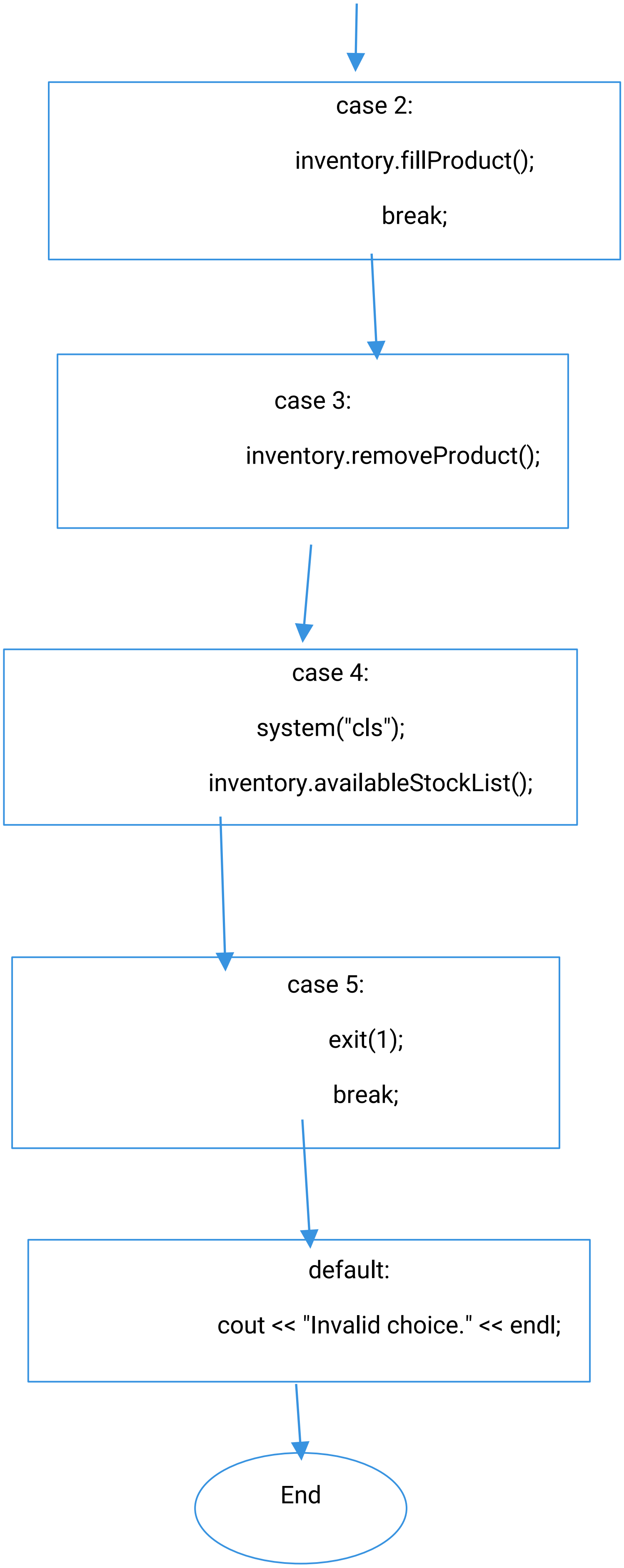
```
Print tempFile << item.stockId << "," <<
ctime(&item.dateTime) << "," <<
item.item.itemName << ","

remove("inventory.csv");

rename("temp.csv", "inventory.csv");

void removeItem(const Stock &item)

ifstream file("inventory.csv");

ofstream tempFile("temp.csv");

string stockId, dateTime, name, quantity, price

while (getline(file,
stockId, ','))
```


getline(file, dateTime, ',');
getline(file, name, ',');
getline(file, quantity, ',
getline(file, price);
if (stockId != item.stockId)
Print tempFile << stockId << "," << dateTime << "," << name << "," << quantity << "," << price << endl;
remove("inventory.csv");
rename("temp.csv", "inventory.csv");


time_t parseDateTime(const string &dateTimeStr)
struct tm tm;
strptime(dateTimeStr.c_str(), "%a %b %d %H:%M:%S %Y", &tm);
return mktime(&tm);
void displayInventory()
int main()
Print WELCOME TO INVENTORY
Print "Enter any key: ";
Print "1. Add New Product"


Print "2. Fill Product"
Print < "3. Remove Product"
Print 4. Available Stock List"
Print 5. Exit "
Print Enter your choice: "
Switch(choice)

```
case 1:
        inventory.addNewProduct();
        break;
```



```
case 2:
        inventory.fillProduct();
                break;

case 3:
        inventory.removeProduct();

case 4:
        system("cls");
        inventory.availableStockList();

case 5:
        exit(1);
        break;

default:
        cout << "Invalid choice." << endl;
```