1. **Program Statement**

This program keeps an inventory for a hardware store four ways; using an unsorted linked list, sorted linked list, unsorted array, and sorted array. User is able to add, update, delete, or list a record along with all the tools that are currently in the inventory.

1. **Requirements**
   1. **Assumptions**
      1. Any words separated by spaces are stored as underscore
      2. User types in an integer for record number and quantity
      3. User types in letters and underscores only for the name
         1. Length of name is less than or equal to 20
      4. User types in a double for cost without any dollar sign in front
      5. Cost is in $
   2. **Specifications**
      1. Menu of four ways to keep inventory (main menu)
         1. Unsorted linked list
            1. Use linear search

Add a record

Update a record

Delete a record

List a record

List ALL tools

Quit

* + - 1. Sorted linked list
         1. Use linear search

Add a record

Update a record

Delete a record

List a record

List ALL tools

Quit

* + - 1. Unsorted array
         1. Use linear search

Add a record

Update a record

Delete a record

List a record

List ALL tools

Quit

* + - 1. Sorted Array
         1. Use linear search

Add a record

Update a record

Delete a record

List a record

List ALL tools

Quit

* + - 1. Quit
    1. Output file that contains the number of comparisons necessary to perform each operations
    2. Struct datatype for each ways being coded
       1. Record # (int)
       2. Tool name (char[20])
       3. Quantity (int)
       4. Cost (double)
    3. Arrays to be initialized to store 100 records (begin with all empty)
    4. Initial Inventory
       1. Record# 3, Electric\_Sander, Quantity: 7, Cost: 57.00
       2. Record# 4, Wire\_Cutteres, Quantity: 8, Cost: 9.99
       3. Record# 17, Hammer, Quantity: 76, Cost: 11.99
       4. Record# 18, Pliers, Quantity: 15, Cost: 19.99
       5. Record# 24, Jig\_Saw, Quantity: 21, Cost: 11.00
       6. Record# 25, Hand\_Drill, Quantity: 5, Cost: 47.75
       7. Record# 39, Lawn\_mower, Quantity: 3, Cost: 79.50
       8. Record# 40, Vise, Quantity: 8, Cost: 34.89
       9. Record# 56, Power\_saw, Quantity: 18, Cost: 99.99
       10. Record# 57, Level, Quantity: 20, Cost: 8.50
       11. Record# 68, Screwdriver, Quantity: 106, Cost: 6.99
       12. Record# 69, Sharpening\_Stone, Quantity: 85, Cost: 5.50
       13. Record# 77, Sledge\_hammer, Quantity: 11, Cost: 21.50
       14. Record# 78, Mallet, Quantity: 21, Cost: 17.99
       15. Record# 83, Wrench, Quantity: 34, Cost: 7.50
       16. Record# 84, Scraper, Quantity: 50, Cost: 8.99
       17. Record# 90, Roller, Quantity: 12, Cost: 14.50
       18. Record# 94, Router, Quantity: 2, Cost: 149.99

1. **Decomposition Diagram**

|  |  |  |
| --- | --- | --- |
| **Main** | | |
| **Input** | **Process** | **Output** |
| Menu selection of Unsorted Linked List | Add a record of user’s choice (operation for each technique of storing inventory) | Completion message (for each process) |
| Menu selection of Sorted Linked List | Update a record of user’s choice (operation for each technique of storing inventory) | List a record of user’s choice |
| Menu selection of Unsorted Array | Delete a record of user’s choice (operation for each technique of storing inventory) | List ALL tools that are not empty from the inventory |
| Menu selection of Sorted Array | Keep track of number of comparisons to perform each operation (for each operation) | Number of comparison necessary to perform each operation (for each operation) |
| Menu selection to quit | Add, update, delete or print all the data of the given record number from the linked list or array | Thank you message |
| Value for record number to be added, updated, deleted, or printed to screen | Add or update the quantity in the linked list or array |  |
| Value for quantity to be added or updated | Add or update the cost in the linked list or array |  |
| Value for cost to be added or updated |  |  |
| Welcome message |  |  |

1. **Test Strategy**
   1. **Valid Data**
   2. **Invalid Data**
2. **Test Plan Version 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 | User selects a number between 0 and 4 for menu selection of a way to store inventories |  |  |  |  |
| Valid | 2 | User selects a number between 0 and 5 for menu selection of operations |  |  |  |  |
| Valid | 3 | User adds a record number between 1 and 100 that is not previously empty |  |  |  |  |
| Valid | 4 | Add Quantity > 0 |  |  |  |  |
| Valid | 5 | Add Cost > 0.0 |  |  |  |  |
| Invalid | 1 | User selects a number less than 0 for menu selection of a way to store inventories |  |  |  |  |
| Invalid | 2 | User selects a number greater than 4 for menu selection of a way to store inventories |  |  |  |  |
| Invalid | 3 | User selects a number less than 0 for menu selection of operations |  |  |  |  |
| Invalid | 4 | User selects a number greater than 5 for menu selection of operations |  |  |  |  |
| Invalid | 5 | Add quantity < 0 |  |  |  |  |
| Invalid | 6 | Add cost < 0.0 |  |  |  |  |

1. **Initial Algorithm**
   1. Include libraries iostream, string, and fstream along with *using namespace std;*
   2. Make a struct and have following items in it
      1. Record # (int)
      2. Tool name (char[20])
      3. Quantity (int)
      4. Cost (double)
      5. next pointer to point to next location of a linked list
   3. Write a function to print a menu of the ways to keep track of inventory
      1. 1. Unsorted Linked List
      2. 2. Sorted Linked List
      3. 3. Unsorted Array
      4. 4. Sorted Array
      5. 5. Quit
   4. Write a function to print a menu of the operations
      1. 1. Add a record
      2. 2. Update a record
      3. 3. Delete a record
      4. 4. List a record
      5. 5. List ALL tools
      6. 6. Quit
   5. In main()
      1. Welcome message
      2. Create an output file to store the number of comparisons necessary to perform each operation
      3. Make an instance of struct created earlier and set everything to NULL
      4. Create head, temporary, and current nodes to keep track of location of unsorted linked list
      5. Create head, temporary, and current nodes to keep track of location of sorted linked list
      6. Make new node for headUnsorted.
      7. Set tempUnsorted and currUnsorted equal to headUnsorted
      8. Make new node for tempUnsorted and assign it’s struct data to 3 for record number, Electric\_Sander for tool name, 7 for quantity, and 57.00 for cost
      9. Set currUnsorted’s next to point to tempUnsorted
      10. Set currUnsorted to tempUnsorted
      11. Repeat last 3 steps for all the inventory items as specified in specifications and set the last inventory item’s next pointer to null
      12. Repeat the last 6 steps, but this time for sorted linked list, so the names of nodes will be different
      13. Initialize an unsorted array of struct data type with maximum values of 100 that it can hold and initially set everything to null
          1. Index 0 has Record# 3, Electric\_Sander, Quantity: 7, Cost: 57.00
          2. Index 1 has Record# 4, Wire\_Cutteres, Quantity: 8, Cost: 9.99
          3. Repeat last 2 steps for rest of inventory items
      14. Initialize a sorted array of struct data type with maximum values of 100 that is can hold and initially set everything to null
          1. Index 2 has Record# 3, Electric\_Sander, Quantity: 7, Cost: 57.00
          2. Index 3 (record number – 1 ) has Record# 4, Wire\_Cutteres, Quantity: 8, Cost: 9.99
          3. Repeat last 2 steps for the rest of inventory items
      15. (do-while loop) do the following, then loop until user input is 0 for the main menu
          1. Call the function to print a menu for list of ways to keep track of inventory
          2. Take user input and store it in a variable for main menu selection
          3. (do-while loop) do the following, then loop until user input is 0 for the operations menu selection.
          4. (switch-case) Switch comparison: main menu selection
             1. For case 1 (unsorted linked list), call the function to print list of operations

Take user input and store it in a variable for sub menu selection

(switch-case) Switch comparison: sub menu selection

For case 1 (Add a record)

Ask the user for record number and check if it is not already in the inventory. Each time you iterate, set a counter variable of how many times you do the comparison. If not, ask user for tool name, quantity, and cost and store it all in specific variables.

Create a new node of current node and store user given values in appropriate struct elements (record #, tool name, quantity, cost)

Check whether list is empty (head == null). Each time you iterate, set a counter variable of how many times you do the comparison

If it is empty, then set new node’s *next* to NULL and head to point to this node

If it is not empty, then use the temporary node pointer created earlier and initialize with head

Keep moving the temporary node to its next node until it reaches to the last node in the list (until temp->next equals null). Each time you iterate, set a counter variable of how many times you do the comparison

Set temporary’s next equal to the new node with given values in it.

Print successful message if operation is successful

Print number of comparisons to an output file.

For case 2 (Update a record)

Ask the user for record number.

Set temporary node to head and go through each node until it’s next value is null to find the given record number. Each time you iterate, set a counter variable of how many times you do the comparison

If record number is found, ask user for the tool name, quantity, and cost and set these values over previous values

If record number not found, print an error message

Print out successful operation message if it is successful

Print number of comparisons to an output file

For case 3 (Delete a record)

Ask the user for the record number.

Set temporary node and current node to head

Keep moving the temporary until it reaches given record number or to the last node. Each time you iterate, set a counter variable of how many times you do the comparison.

Set current node to temp node every time before moving the temporary node to its next node

If reached last node, print an error message

If found, check whether temporary node is the first node in the list. If it is, then move the head to its next node and delete temporary node

If temporary not first node, then check if is last node in the list. If it is, then set current’s next to null and delete temporary node

If temporary node not first or last node, set current’s next to temporary’s next and delete temporary node

Print out successful operation message

Print number of comparisons to the output file

For case 4 (List a record)

Ask the user for the record number

Set temporary node to head

Go through each list and search for the given record number. Each time you iterate, set a counter variable of how many times you do the comparison

If found, print out the elements in it in organized format

If not found, print out an error message

Print number of comparisons to the output file

For case 5 (list ALL tools)

Set temporary node to head

Got through each list and print out the elements in each linked list in an organized manner

For case 6 (Quit), Thank you message

* + - * 1. For case 2 (sorted linked list), call the function to print list of operations

Take user input and store it in a variable for sub menu selection

(switch-case) Switch comparison: sub menu selection

For case 1 (Add a record)

Ask the user for record number and check if it is not already in the inventory. Each time you iterate, set a counter variable of how many times you do the comparison. If not, ask user for tool name, quantity, and cost and store it all in specific variables.

Create a new node of current node and store user given values in appropriate struct elements (record #, tool name, quantity, cost)

Check whether list is empty (head == null)

If it is empty, then set new node’s *next* to NULL and head to point to this node

If it is not empty, then use the temporary node pointer created earlier and initialize with head

Keep moving the temporary node to its next node until it reaches to the last node in the list (until temp->next equals null). Each time you iterate, set a counter variable of how many times you do the comparison.

Set temporary’s next equal to the new node with given values in it

Print successful message if operation was successful.

Print number of comparisons to the output file

For case 2 (Update a record)

Ask the user for record number.

Set temporary node to head and go through each node until its next value’s record number is greater than given number to find the given record number. Each time you iterate, set a counter variable of how many times you do the comparison

If record number is found, ask user for the tool name, quantity, and cost and set these values over previous values

If record number not found, print an error message

Print out successful operation message if it is successful

Print number of comparisons to the output file

For case 3 (Delete a record)

Ask the user for the record number.

Set temporary node and current node to head

Keep moving the temporary until it reaches given record number or until the next value’s record number is greater than temporary node’s record number. Each time you iterate, set a counter variable of how many times you do the comparison

Set current node to temp node every time before moving the temporary node to its next node

If reached last node, print an error message

If found, check whether temporary node is the first node in the list. If it is, then move the head to its next node and delete temporary node

If temporary not first node, then check if is last node in the list. If it is, then set current’s next to null and delete temporary node

If temporary node not first or last node, set current’s next to temporary’s next and delete temporary node

Print out successful operation message

Print number of comparisons to the output file

For case 4 (List a record)

Ask the user for the record number

Set temporary node to head

Go through the list until temporary node’s next value for record number is greater than given record number and search for the given record number. Each time you iterate, set a counter variable of how many times you do the comparison

If found, print out the elements in it in organized format

If not found, print out an error message

Print number of comparisons to the output file

For case 5 (list ALL tools)

Set temporary node to head

Got through each list and print out the elements in each linked list in an organized manner

For case 6 (Quit), Thank you message

* + - * 1. For case 3 (unsorted array), call the function to print list of operations

Take user input and store it in a variable for sub menu selection

(switch-case) Switch comparison: sub menu selection

For case 1 (Add a record)

Ask user for the record number

Go through each index and stop if an index that is null is found or a record number matches the given record number. Each time you iterate, set a counter variable of how many times you do the comparison

If index that is null found with no repetitive record number, ask the user for the tool name, quantity, and cost and add these data in the struct in array.

If given record number already exists in the array, print out an error message

Print successful message if operation is successful

Print number of comparisons to the output file

For case 2 (Update a record)

Ask the user for the record number

Go through each index and stop if an index matches the record number – 1. Each time you iterate, set a counter variable of how many times you do the comparison

If there is no match, print out an error message of not finding the record number

If found, ask user for tool name, quantity, and cost and change the elements current index to these values

Print out successful message if operation is successful

Print number of comparisons to the output file

For case 3 (Delete a record)

Ask the user for the record number

Go through each index and stop if an index matches the record number – 1. Each time you iterate, set a counter variable of how many times you do the comparison

If there is no match, print out an error message of not finding the record number

If found, set every element to null

Print out successful message if operation is successful

Print number of comparisons to the output file

For case 4 (List a record)

Ask user for the record number

Go through each index and stop if an index matches the record number – 1. Each time you iterate, set a counter variable of how many times you do the comparison

If there is no match, print out an error message of not finding the record number

If found, print values of all the elements in this index in organized manner

Print number of comparisons to the output file

For case 5 (list ALL tools)

Go through each index and print all the elements in the array in organized manner

For case 6 (Quit), Thank you message

* + - * 1. For case 4 (sorted array), call the function to print list of operations

Take user input and store it in a variable for sub menu selection

(switch-case) Switch comparison: sub menu selection

For case 1 (Add a record)

Ask user for the record number

Check if index number of given record number – 1 is null. Each time you iterate, set a counter variable of how many times you do the comparison

If not null, print an error message of record already exists

If null, ask user for tool name, quantity, and cost and add these values to the struct in current index

Print successful message if operation was successful

Print number of comparisons to the output file

For case 2 (Update a record)

Ask user for the record number

Check if index number of given record number – 1 is not null. Each time you iterate, set a counter variable of how many times you do the comparison

If null, print an error message of record does not exist

If record exists in this index, ask user for tool name, quantity, and cost and replace previous values to these values in the struct in current index.

Print successful message if operation was successful

Print number of comparisons to the output file

For case 3 (Delete a record)

Ask user for record number

Check if index number of given record number – 1 is not null

If null, print an error message of record number being already empty

If not null, then set each value in the current index to null

Print successful message if operation was successful

For case 4 (List a record)

Ask user for record number

Check if index number of given record number – 1 is not null

If null, print an error message of record number being empty

If not null, print each data in this index in organized format

For case 5 (list ALL tools)

Go through each index of array that is not null and print each data in each index in an organized manner

For case 6 (Quit), Thank you message

* + - * 1. For case 0 (quit), Thank you message

1. **Test Plan Version 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 | User selects a number between 0 and 4 for menu selection of a way to store inventories | 3 | Shows the menu for operations for an unsorted array |  |  |
| Valid | 2 | User selects a number between 0 and 5 for menu selection of operations | User selects #5 after selecting 3 for unsorted array | Prints all the data in the unsorted array in organized format |  |  |
| Valid | 3 | User adds a record number between 1 and 100 that is not previously empty | User adds record number 2. Tool name: measure\_tape  Quantity: 10  Cost: 15.00 | Successful operation message |  |  |
| Valid | 4 | Add Quantity > 0 | Quantity 10 for record number 1 in an unsorted array. Tool name: push\_pin. Cost: 1.99 | Successful operation message |  |  |
| Valid | 5 | Add Cost > 0.0 | Cost: 19.99 for record number 5 in an unsorted array. Tool name: gloves. Quantity: 19 | Successful operation message |  |  |
| Invalid | 1 | User selects a number less than 0 for menu selection of a way to store inventories | -5 | Error message saying invalid input |  |  |
| Invalid | 2 | User selects a number greater than 4 for menu selection of a way to store inventories | 10 | Error message saying invalid input |  |  |
| Invalid | 3 | User selects a number less than 0 for menu selection of operations | -4 | Error message saying invalid input |  |  |
| Invalid | 4 | User selects a number greater than 5 for menu selection of operations | 15 | Error message saying invalid input |  |  |
| Invalid | 5 | Add quantity < 0 | Unsorted array, record # 6, tool name: glue\_gun, quantity -9, cost: 3.99 | Error message saying invalid quantity |  |  |
| Invalid | 6 | Add cost < 0.0 | Unsorted array, record # 7, tool name: bolt, cost: -0.99 | Error message saying invalid cost |  |  |

1. **Code**
2. **Updated Algorithm**
3. **Test Plan Version 3**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 |  |  |  |  |  |

1. **Screenshots**
2. **Status**