DATE: PAGE:

Freddy Jenson
RMCA A 2020-2022

# Merging

Step 1 Start
Step 2 Declare the variables
Step 3 Read the size of first array
Step 4 Read elements of first array in second order
Step 5 Read the size of second array
Step 6 Read the elements of second array in sorted order
Step 7 Repeat steps 8 and 9, while $i<m$ & $j<n$
Step 8 Check if $a[i] >= b[j]$, then $c[k++] = b[j++]$
Step 9 Else $c[k++] = a[i++]$
Step 10 Repeat Step 11 while $i<m$
Step 11 $c[k++] = a[j++]$
Step 12 Repeat step 13 while $j<n$
Step 13 $c[k++] = b[j++]$
Step 14 Print the first array
Step 15 Print the second array
Step 16 Print the sorted merged array
Step 17 End

DATE: PAGE:

Output

Size of first Array
2
Enter value in sorted order
3
6
Size of second Array
4
Enter value in sorted order
4
5
7
8
Array A:
3 6
Array B
4 5 7 8

Merged Array
3 4 5 6 7 8

# Stack Operations

Step 1 Start

Step 2 Declare the node and the required variables

Step 3 Declare the functions for push, pop, display and search an element

Step 4 Read the choice from user

Step 5 If the user choose to push an element. then read the element to be pushed & call the function to push the element by passing the value to the function

Step 5.1 Declare the newnode & allocate memory for the newnode

Step 5.2 Set newnode → data = value

Step 5.3 Check if top == null, then set newnode → next = null

Step 5.4 Set newnode → next = top

Step 5.5 Set top = newnode & then print insertion is successfull

Step 5.6 If user choose to pop an element from the stack, then call the function to pop the element

Step 6.1 Check if top == null, then print stack is empty

Step 6.2 Else declare a pointer variable temp and initialize it to top

Step 6.3 Print the element that being deleted

Step 6.4 Set temp = temp → next

Step 6.5 Free the temp

Step 7 If the user choose the display then call the function to display the element in the stack

Step 7.1 Check if top == null then print stack is empty

Step 7.2 Else declare a pointer variable temp & initialize it to top

Step 7.3 Repeat steps below while temp → next != null

Step 7.4 Print temp → data

Step 7.5 Set temp = temp → next

Step 8 If the user choose to search an element from the stack then call the function to search an element

Step 8.1 Declare a pointer variable ptr and other necessary variable

Step 8.2 Initialize ptr = top
Step 8.3 Check if ptr = null then print stack empty
Step 8.4 Else read the element to be searched
Step 8.5 Repeat step 8.6 to 8.8 while ptr != null
Step 8.6 Check if ptr → data == item, then print element found and to be located and set flag = 1
Step 8.7 Else set flag = 0
Step 8.8 Increment i by 1 and set ptr = ptr → next
Step 8.9 Check if flag = 0, then print the element not found
Step 9 End

DATE: PAGE:

## Output

Menu
1. Push
2. Pop
3. display
4. Search
5. Exit

Enter the choice : 1
Enter the element to be inserted : 7
Insertion is successful

Menu
1. Push

Enter the choice : 1
Enter the element to be inserted : 4
Insertion is successful

Menu
1. Push
Enter the choice : 1
Enter the element to be inserted : 10
Insertion is successful

DATE: PAGE:

Menu

2. Pop

Enter the choice: 2

Element deleted: 10

Menu

3. Display

Enter the choice: 3

4 → 2 → null

Menu

4. search

Enter the choice: 4

Enter the term to be searched: 2

item found at location: 2

Menu

5. exit

enter the choice: 5

# Circular Queue Operation

Step 1 Start

Step 2 Declare the queue and other variables

Step 3 Declare the functions for enqueue, dequeue, search and display

Step 4 Read the choice from the user

Step 5 If the user choose the choice enqueue then, read the element to be inserted from the user and call the enqueue function by passing the value

Step 5.1 Check if front == -1 && rear == -1, then set front = 0, rear = 0 and set queue[rear] = element

Step 5.2 Else if rear + 1 % max == front or front = rear + 1, then print Queue is overflow

Step 5.3 Else set rear = rear + 1 % max and set queue[rear] = element

Step 6 If the user choice is the option dequeue then call the function dequeue

Step 6.1 Check if front == -1 and rear == -1 then print Queue is overflow

Step 6.2 Else check if front == rear then print the element is to be deleted Then set front = -1 and rear = -1.

Step 6.3 Else print the element to be dequeued set front = front + 1 % max

Step 7 If the user choice is to display the queue then call the function display

Step 7-1 Check if front = -1 and rear = -1, then print Queue is empty

Step 7-2 Else repeat the step 7-3 while i <= rear

Step 7-3 Print queue[i] and set i = i + 1 % max

Step 8 If the user choose the search then call the function to search an element in the queue

Step 8.1 Read the element to be searched in the queue

Step 8.2 Check if item == queue[i] then print item found and its position and increment i by 1

Step 8.3 Check if c == 0, then print item not found

Step 9 End

DATE: PAGE:

Output

Menu

1. Insert 2. Delete 3. Display 4. Search 5. Exit

Enter the choice: 1

Enter the number to insert: 10

Menu

Enter choice: 1

Enter the number to insert: 20

Menu

Enter the choice: 1

Enter the number to insert: 30

Menu

Enter the choice: 3

10, 20, 30

Menu

Enter the choice: 4

Enter the element to be searched: 30

Item found at location 3

Menu

Enter the choice: 2

10 was deleted!

Menu

Enter the choice: 5

# Doubly Linked List Operation

Step 1 Start

Step 2 Declare a structure and related variables

Step 3 Declare functions to create a node, insert a node in the begining, at the end and given position, display the list and search an element in the list

Step 4 Define function to create a node, declare required variables.

Step 4.1 Set memory allocated to the node = temp, then set temp->pre = null and tempt->next = null

Step 4.2 Read the value to be inserted to the node

Step 4.3 Set temp->n = data and increment count by 1

Step 5 Read the choice from the user to perform different operation on the list

Step 6 If the user choose to perform insertion operation at the begining then call the function

to perform the insertion

Step 6.1 Check if head == null then call the function to create a node, perform step 4 to step 4.3

Step 6.2 Set head = temp and temp1 = head

Step 6.3 Else call the function to create a node, perform step 4 to 4.3, then set temp → next = head, set head → prev = temp and head = temp

Step 7 If the user choice is to perform insertion at the end of the list, then call the function to perform insertion at the end

Step 7.1 Check if head == null then call the function to create a newnode then set temp = head and then set head = temp1

Step 7.2 Else call the function to create a newnode then set temp1 → next = temp, temp → prev = temp1 and temp1 = temp

Step 8 If the user choose to perform insertion in the list at any position then call the function to perform the insertion operation

Step 8.1 Declare the necessary variable

Step 8.2 Read the position where the node need to be inserted, set temp2 = head

Step 8.3 Check if pos < 1 or pos >= count + 1 then print the position is out of range

Step 8.4 Check if head == null and pos = 1 then print Empty List cannot insert other than first position

Step 8.5 Check if head == null and pos = 1 then call the function to create Newnode then set temp = head and head = temp1

Step 8.6 While i < pos, then set temp2 = temp->next then increment i by 1

Step 8.7 Call the function to create a newnode and then set temp->prev = temp2, temp->next = temp2->next-> prev = temp, temp2->next = temp

Step 9 If the user choose to perform deletion operation is the list then all the function to perform the deletion operation

Step 9.1 Declare the necessary variables

Step 9.2 Read the position where node need to be deleted, set temp2 = head

Step 9.3 Check if pos < 1 or pos >= count + 1, then print position out of range

Step 9.4 Check if head == null, then print the list is empty

Step 9.5 while i < pos then temp2 = temp2 → next and increment i by 1

Step 9.6 Check if i = 1, then check if temp2 → next == null, then print node deleted Free (temp2), set temp2 = head = null

Step 9.7 Check if temp2 → next == null then temp2 → prev → next = null then free (temp2), then print Node deleted

Step 9.8 temp2 → next → prev = temp2 → prev then print check if i != 1, then temp2 → prev → next = temp2 → next

Step 9.9 Check if i = 1, then head = temp2 → next then print node deleted then free temp2 and decrement count by 1.

Step 10 If the user choose to perform the display operation then call the function to display the list

Step 10: Set temp2 = h
Step 10-2 Check if temp2 = null, then print list is empty
Step 10-3 While temp2 → next != null then print temp2 → n then temp2 = temp2 → next
Step 11 If the user choose to perform the search operation then call the function to perform search operations
Step 11-1 Declare necessary variables
Step 11-2 Set temp2 = head
Step 11-3 Check if temp2 == null then print the list is empty
Step 11-4 Read the value to be searched
Step 11-5 While temp2 != null, then check if temp2 → n == data, then print element found at position count+1
Step 11-6 Else set temp2 = temp2 → next and increment count by 1
Step 11-7 Print element not found in the list
Step 12 End

DATE: PAGE:

Output

1. Insert at beginning
2. Insert at end
3. Insert at position
4. Delete
5. Display
6. Search
7. Exit

Enter choice: 1
Enter the value to node: 5
Menu
Enter choice: 1
Enter the value to node: 10
Enter choice: 2
Enter value to node: 2
Enter choice: 3
Enter the position to be inserted: 2
Enter the value to node: 13

Menu
Enter choice: 5

Linked list element from beginning
10 13 5 2

DATE: PAGE:

Menu
Enter choice: 4
Enter the position to be deleted: 2
Node deleted

Menu
Enter choice: 6
Enter value to search: 10
Element found in 1 position

Menu
Enter choice: 7

# Set Operations

Step 1 Start
Step 2 Declare necessary variables
Step 3 Read the choice from the user to perform set operation
Step 4 If the user choose to perform union
step 4.1 Read the cardinality of two sets
Step 4.2 Check if m!=n, then print union cannot be performed
Step 4.3 Else read the elements of both sets
Step 4.4 Repeat the step 4.5 to 4.7 until i<m
Step 4.5 c[i] = A[i]|B[i]
Step 4.6 Print c[i]
Step 4.7 Increment i by 1
Step 5 Read the choice from user to perform intersection
Step 5.1 Read the cardinality of two sets
Step 5.2 Check if m!=n, then print cannot perform intersection
Step 5.3 Else read the elements of both sets
Step 5.4 Repeat steps 5.5 to 5.7 until i<m
Steps 5.5 c[i] = A[i]|B[j]
Step 5.6 Print c[i]
Step 5.7 Increment i by 1

Step 6 If the user choose to perform set difference operation

Step 6-1 Read the cardinality of two sets

Step 6-2 Check if m!=n, then print cannot perform set difference operation

Step 6-3 Else read the elements of both sets

Step 6-4 Repeat the steps 6-5 to 6-8 until i<n

Step 6-5 Check if A[i]=0, then c[i]=0

Step 6-6 Else if B[i]==1, then c[i]=0

Step 6-7 Else c[i]=1

Step 6-8 Increment i by 1

Step 7 Repeat the steps 7-1 and 7-2 until i<=m

Step 7-1 print c[i]

Step 7-2 Increment i by 1

Step 8 If End

DATE: PAGE:

Output

Press 1 for union
Press 2 for intersection
Press 3 for subtraction
Press 4 for exit
enter choice 1
Enter the size of set 1
3
Enter the element of set 1
1 2 3
Enter the size of set 2
2 3
Enter the element of set 2
2 3
Union: 1 2 3
Menu
Enter your choice: 2
enter the size of set 1
3
enter the element of set 1
1 2 3
Enter the size of set 2
2

DATE: PAGE:

Enter the element of set 2
3 4
menu
Enter the size of set 1
3
Enter the element of set 4
1 2 3
Enter the size of set 2
2
Enter the element of set 2
3 2

difference : 1
menu

# Binary Search Tree

Step 1 Start

Step 2 Declare a structure and structure pointers for insertion, deletion and search operation and also declare a function for inorder traversal

Step 3 Declare a pointer as root and also the required variable

Step 4 Read the choice from the user to perform insertion, deletion, searching and inorder traversal

Step 5 If the user choose to perform insertion operation then read the value which is to be inserted to the tree from the user

Step 5.1 Pass the value to the insert pointer and also the root pointer

Step 5.2 Check if ! root then allocate memory for the root

Step 5.3 Set the value to the info part of the root and then set left and right part of the root to null and return root

Step 5.4 Check if root → info > x then call the insert pointer to insert to left of the root

Step 5.5 Check if root → info < x then call the insert pointer to insert to the right of the root

Step 5.6 Return the root

Step 6 If the user choose to perform deletion operation then read the element to be deleted from the tree, pass the root pointer and the item to the delete pointer.

Step 6.1 Check if not ptr then print node not found.

Step 6.2 Else if ptr → info < x then call delete pointer by passing the right pointer and the item

Step 6.3 Else if ptr → info > x then call delete pointer by passing the left pointer and the item

Step 6.4 Check if ptr → info == item then check if ptr → left == ptr → right then free ptr and return null

Step 6.5 Else if ptr → left == null then set P1 . ptr → right and free ptr, return P1

Step 6.6 Else if ptr → right == null then set P1 = Ptr → left and free ptr, return P1

Step 6.7 Else set P1 = ptr → right and P2 = ptr → right

Step 6.8 While P1 → left not equal to null, set P1 → left. ptr → left and free ptr. return P2

Step 6.9 Return Ptr

Step 7 If the user choose to perform search operation the call the pointer to perform search operation

Step 7.1 Declare the necessary pointers and variables

Step 7.2 Read the element to be searched

Step 7.3 While ptr check if item > ptr → info then ptr = ptr → right

Step 7.4 Else if item < ptr → info then ptr = ptr → left

Step 7.5 Else break

Step 7.6 Check if ptr then print that the element is found

Step 7.7 Else print element not found in tree and return root

Step 8 If the user choose to perform traversal then call the traversal function and pass the root pointers

Step 8.1 If root not equal to null recursivly call the functions by passing root → left

Step 8.2 Print root → info

Step 8.3 Call the traversal function recursively by passing root → right

Step 9 End

DATE: PAGE:

Output

1. Insert in Binary tree
2. Delete from Binary tree
3. Inorder traversal of Binary Tree
4. Search
5. Exit

Enter choice: 1
Enter new element 20
root is 20
Menu

Enter choice: 1
Enter new element: 25
Inorder traversal of binary tree is: 20 25

Menu
Enter choice: 4
Search operation in binary tree
Enter the element to be searched: 5
Element, 25 is found.