1. **Write a menu driven program in C to perform array operations (Insertion, Deletion, Reversing, Searching, Sorting, Modifying, Displaying) using user defined functions.**

**Program: prg1.c**

#include <stdio.h>

#include <stdlib.h>

#define max\_size 20

void display(int[], int);

int insertAtPosition(int[], int, int, int);

int deleteAtPosition(int[], int, int);

void linearSearch(int[], int, int);

void binarySearch(int[], int, int);

void sortArray(int[], int);

void reverse(int[], int);

int main()

{

int arr[max\_size], size, i, choice, data, pos;

printf("\n\t\t\t\t\t-------ARRAY OPERATIONS-------\n\n");

do

{

printf("How many elements you want to enter: ", max\_size);

scanf("%d", &size);

if (size < 0 || size > 20)

printf("The number of elements must be smaller than or equals to %d !!!Try again\n\n", max\_size);

} while (size < 0 || size > 20);

printf("\nEnter data one by one for array elements:\n");

for (i = 0; i < size; i++)

{

printf("\tArr[%d] : ", i);

scanf("%d", &arr[i]);

}

while (1)

{

printf("\nPress Enter to continue..... ");

fflush(stdin);

getchar();

system("cls");

printf("\n\t: AVAILAVLE OPTIONS :\n\n");

printf(" 1. Insert an Element\n 2. Delete an Element\n 3. Search for a Element\n");

printf(" 4. Sort the array\n 5. Reverse the array\n 6. Display the whole array\n 0. Exit\n");

printf("\nEnter your choice : ");

scanf("%d", &choice);

switch (choice)

{

case 0:

printf("\n\tTHANK YOU\n");

exit(0);

case 1:

system("cls");

printf("\n\t#ELEMENT INSERTION#\n\n");

do

{

printf("Enter the position you want to insert the data (1 to %d) : ", size);

scanf("%d", &pos);

if (pos < 1 || pos > size)

printf("Error! : Invalid Position. Try Again\n\n");

} while (pos < 1 || pos > size);

printf("Enter the new element you want to insert : ");

scanf("%d", &data);

display(arr, size);

size = insertAtPosition(arr, size, pos, data);

printf(">> New element %d successfully entered at position %d\n\n", data, pos);

display(arr, size);

break;

case 2:

// ----Deletion----

system("cls");

printf("\n\t#ELEMENT DELETION#\n\n");

do

{

printf("Enter the position of the data you want to delete (1 to %d) : ", size);

scanf("%d", &pos);

if (pos < 1 || pos > size)

printf("Error! : Invalid Position. Try Again\n\n");

} while (pos < 1 || pos > size);

display(arr, size);

size = deleteAtPosition(arr, size, pos);

printf(">> Element successfuly deleted from position %d\n\n", pos);

display(arr, size);

break;

case 3:

// -----Search a Element-----

system("cls");

printf("\n\t#SEARCH ELEMENT#\n\n");

printf("Which type of search you want ?\n");

printf(" 1. Linear Search\n");

printf(" 2. Binary Search\n=> ");

scanf("%d", &choice);

printf("Enter the element you want to search : ");

scanf("%d", &data);

display(arr, size);

if (choice == 1)

linearSearch(arr, size, data);

else if (choice == 2)

binarySearch(arr, size, data);

break;

case 4:

// -----Sort array-----

system("cls");

printf("\n\t#SORT ARRAY IN ASSENDING ORDER#\n\n");

printf("Before Sorting : ");

display(arr, size);

sortArray(arr, size);

printf("After Sorting : ");

display(arr, size);

break;

case 5:

// -----Reverse-----

system("cls");

printf("\n\tARRAY REVERSE\n\n");

printf("\nThe array before reverse: \n");

display(arr, size);

reverse(arr, size);

printf("\nThe array after reverse: \n");

display(arr, size);

break;

case 6:

display(arr, size);

break;

default:

printf("\n\tERROR! Wrong Choice!\t");

}

}

return 0;

}

void display(int arr[], int size)

{

int i, pos;

printf("\nThe Array is : \n\t");

printf("--------------------------------------------------\n\t");

for (i = 0; i < size; i++)

{

printf(" %d |", arr[i]);

}

printf("\n\t--------------------------------------------------\n\t");

for (i = 0; i < size; i++)

{

printf("\n\tArr[%d] = %d", i, arr[i]);

}

printf("\n");

}

int insertAtPosition(int arr[], int size, int pos, int item)

{

int i, temp; for (i = size - 1; i >= pos - 1; i--)

{

arr[i + 1] = arr[i];

arr[pos - 1] = item;

return size + 1;

}

int deleteAtPosition(int arr[], int size, int pos)

{

int i, temp;

for (i = pos - 1; i < size - 1; i++)

{

arr[i] = arr[i + 1];

}

return size - 1;

}

void reverse(int arr[], int size)

{

int i, upto, temp;

if (size % 2 == 0)

upto = size / 2 - 1;

else

upto = size / 2;

for (i = 0; i <= upto; i++)

{

temp = arr[i];

arr[i] = arr[size - i - 1];

arr[size - i - 1] = temp;

}

}

void linearSearch(int arr[], int size, int item)

{

int i, count = 0;

printf("\n[ LINEAR SEARCHING FOR = %d ]\n\n");

for (i = 0; i < size; i++)

{

if (arr[i] == item)

{

printf(">> Found at position = %d\n", i + 1);

count++;

}

}

if (count == 0)

printf("Element not found in the array!");

}

void binarySearch(int arr[], int size, int item)

{

int i, count = 0, beg, mid, end;

printf("\n[ BINARY SEARCHING FOR = %d ]\n\n");

printf("\n> Sorting the array before starting binary search...");

sortArray(arr, size);

display(arr, size);

beg = 0;

arr[pos - 1] = item;

return size + 1;

}

int deleteAtPosition(int arr[], int size, int pos)

{

int i, temp;

for (i = pos - 1; i < size - 1; i++)

{

arr[i] = arr[i + 1];

}

return size - 1;

}

void reverse(int arr[], int size)

{

int i, upto, temp;

if (size % 2 == 0)

upto = size / 2 - 1;

else

upto = size / 2;

for (i = 0; i <= upto; i++)

{

temp = arr[i];

arr[i] = arr[size - i - 1];

arr[size - i - 1] = temp;

}

}

void linearSearch(int arr[], int size, int item)

{

int i, count = 0;

printf("\n[ LINEAR SEARCHING FOR = %d ]\n\n");

for (i = 0; i < size; i++)

{

if (arr[i] == item)

{

printf(">> Found at position = %d\n", i + 1);

count++;

}

}

if (count == 0)

printf("Element not found in the array!");

}

void binarySearch(int arr[], int size, int item)

{

int i, count = 0, beg, mid, end;

printf("\n[ BINARY SEARCHING FOR = %d ]\n\n");

printf("\n> Sorting the array before starting binary search...");

sortArray(arr, size);

display(arr, size);

beg = 0;

end = size - 1;

mid = (beg + end) / 2;

mid = (beg + end) / 2;

printf("\nbeg = %d , mid = %d , end = %d\n", beg, mid, end);

while ((item != arr[mid]) && (beg <= end))

{

mid = (beg + end) / 2;

if (arr[mid] < item)

beg = mid + 1;

else if (arr[mid] > item)

end = mid - 1;

else if (arr[mid] == item)

{

printf("Item found at = %d\n", mid);

break;

}

}

if (arr[mid] == item)

printf("Item %d found at = %d\n", item, mid);

else

printf("Item %d not found in the array\n", item);

}

void sortArray(int arr[], int size)

{

int i, j, temp;

for (i = 0; i < size - 1; i++)

{

for (j = 0; j < size - i - 1; j++)

{

if (arr[j] > arr[j + 1])

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

**OUTPUT:**

-------ARRAY OPERATIONS-------

How many elements you want to enter: 4

Enter data one by one for array elements:

Arr[0] : 10

Arr[1] : 25

Arr[2] : 37

Arr[3] : 17

Press Enter to continue.....

: AVAILAVLE OPTIONS :

1. Insert an Element

2. Delete an Element

3. Search for a Element

4. Sort the array

5. Reverse the array

6. Display the whole array

0. Exit

Enter your choice : 1

#ELEMENT INSERTION#

Enter the position you want to insert the data (1 to 4) : 3

Enter the new element you want to insert : 20

The Array is :

--------------------------------------------------

10 | 25 | 37 | 17 |

--------------------------------------------------

Arr[0] = 10

Arr[1] = 25

Arr[2] = 37

Arr[3] = 17

>> New element 20 successfully entered at position 3

The Array is :

--------------------------------------------------

10 | 25 | 20 | 37 | 17 |

--------------------------------------------------

Arr[0] = 10

Arr[1] = 25

Arr[2] = 20

Arr[3] = 37

Arr[4] = 17

Press Enter to continue..…

: AVAILAVLE OPTIONS :

1. Insert an Element

2. Delete an Element

3. Search for a Element

4. Sort the array

5. Reverse the array

6. Display the whole array

0. Exit

Enter your choice : 2

#ELEMENT DELETION#

Enter the position of the data you want to delete (1 to 5) : 1

The Array is :

--------------------------------------------------

10 | 25 | 20 | 37 | 17 |

--------------------------------------------------

Arr[0] = 10

Arr[1] = 25

Arr[2] = 20

Arr[3] = 37

Arr[4] = 17

>> Element successfuly deleted from position 1

The Array is :

--------------------------------------------------

25 | 20 | 37 | 17 |

--------------------------------------------------

Arr[0] = 25

Arr[1] = 20

Arr[2] = 37

Arr[3] = 17

Press Enter to continue..…

: AVAILAVLE OPTIONS :

1. Insert an Element

2. Delete an Element

3. Search for a Element

4. Sort the array

5. Reverse the array

6. Display the whole array

0. Exit

Enter your choice : 3

#SEARCH ELEMENT#

Which type of search you want ?

1. Linear Search

2. Binary Search

=> 1

Enter the element you want to search : 17

The Array is :

--------------------------------------------------

25 | 20 | 37 | 17 |

--------------------------------------------------

Arr[0] = 25

Arr[1] = 20

Arr[2] = 37

Arr[3] = 17

LINEAR SEARCHING FOR = 4

>> Found at position = 4

Press Enter to continue..…

#SEARCH ELEMENT#

Which type of search you want ?

1. Linear Search

2. Binary Search

=> 1

Enter the element you want to search : 10

The Array is :

--------------------------------------------------

25 | 20 | 37 | 17 |

--------------------------------------------------

Arr[0] = 25

Arr[1] = 20

Arr[2] = 37

Arr[3] = 17

LINEAR SEARCHING FOR = 4

Element not found in the array!

Press Enter to continue..…

#SEARCH ELEMENT#

Which type of search you want ?

1. Linear Search

2. Binary Search

=> 2

Enter the element you want to search : 20

The Array is :

--------------------------------------------------

25 | 20 | 37 | 17 |

--------------------------------------------------

Arr[0] = 25

Arr[1] = 20

Arr[2] = 37

Arr[3] = 17

[ BINARY SEARCHING FOR = 4 ]

> Sorting the array before starting binary search...

The Array is :

--------------------------------------------------

17 | 20 | 25 | 37 |

--------------------------------------------------

Arr[0] = 17

Arr[1] = 20

Arr[2] = 25

Arr[3] = 37

Item 20 found at index 1

Press Enter to continue.....

: AVAILAVLE OPTIONS :

1. Insert an Element

2. Delete an Element

3. Search for a Element

4. Sort the array

5. Reverse the array

6. Display the whole array

0. Exit

Enter your choice : 4

#SORT ARRAY IN ASSENDING ORDER#

Before Sorting :

The Array is :

--------------------------------------------------

17 | 20 | 25 | 37 |

--------------------------------------------------

Arr[0] = 17

Arr[1] = 20

Arr[2] = 25

Arr[3] = 37

After Sorting :

The Array is :

--------------------------------------------------

17 | 20 | 25 | 37 |

--------------------------------------------------

Arr[0] = 17

Arr[1] = 20

Arr[2] = 25

Arr[3] = 37

Press Enter to continue..…

: AVAILAVLE OPTIONS :

1. Insert an Element

2. Delete an Element

3. Search for a Element

4. Sort the array

5. Reverse the array

6. Display the whole array

0. Exit

Enter your choice : 5

#ARRAY REVERSE#

The array before reverse:

The Array is :

--------------------------------------------------

17 | 20 | 25 | 37 |

--------------------------------------------------

Arr[0] = 17

Arr[1] = 20

Arr[2] = 25

Arr[3] = 37

The array after reverse:

The Array is :

--------------------------------------------------

37 | 25 | 20 | 17 |

--------------------------------------------------

Arr[0] = 37

Arr[1] = 25

Arr[2] = 20

Arr[3] = 17

Press Enter to continue.....

: AVAILAVLE OPTIONS :

1. Insert an Element

2. Delete an Element

3. Search for a Element

4. Sort the array

5. Reverse the array

6. Display the whole array

0. Exit

Enter your choice : 6

The Array is :

--------------------------------------------------

37 | 25 | 20 | 17 |

--------------------------------------------------

Arr[0] = 37

Arr[1] = 25

Arr[2] = 20

Arr[3] = 17

Press Enter to continue..…

: AVAILAVLE OPTIONS :

1. Insert an Element

2. Delete an Element

3. Search for a Element

4. Sort the array

5. Reverse the array

6. Display the whole array

0. Exit

Enter your choice : 0

THANK YOU

--------------------------------

Process exited after 596.1 seconds with return value 0

Press any key to continue . . .