Data Structure

[2022]

[01.09.2022]

Techno India College of Technology
Authored by: Abu Taha Md Farhad
Roll: 31401221052
Reg No: 213141001210016 OF 2021-22
Stream: BCA 2nd year
Subject: Data Structure Lab [BCAC393]

Code:

Insertion:

array_insert_beginning.c:

```c
int insert_at_beginning(int *arr, int* n, int val){
    for (int i = *n; i >= 1; i--)
    {
        *(arr+i) = *(arr+(i-1));
    }

    *arr = val;
    *n = (*n)+1;
}
```

array_insert_last.c:

```c
int insert_at_last(int *arr, int* n, int val){
    *(arr+*n) = val;
    *n = (*n)+1;
}
```

array_insert_position.c:

```c
int insert_at_position(int *arr, int* n, int val, int pos){
    for (int i = *n; i >= pos; i--)
    {
        *(arr+i) = *(arr+(i-1));
    }

    *(arr+pos) = val;
    *n = (*n)+1;
}
```

Deleteion:

array_delete_beginning.c:

```c
int delete_from_beginning(int *arr, int* n) {
    for (int i = 0; i < *n-1; i++)
    {
        *(arr+i) = *(arr+(i+1));
    }
    *(arr+(*n-1)) = 0;
    *n = (*n)-1;
}
```

array_delete_last.c

```c
int delete_from_last(int *arr, int* n){
    *(arr+(*n-1)) = 0;
    *n = (*n)-1;
}
```

array_delete_position.c:

```c
int delete_from_position(int *arr, int* n, int pos){
    for (int i = pos; i < *n-1; i++)
    {
        *(arr+i) = *(arr+(i+1));
    }

    *(arr+(*n-1)) = 0;
    *n = (*n)-1;
}
```

Searching:

array_search_linear.c:

```c
int search_linear(int *arr, int n, int search_val) {
    for (int i = 0; i < n; i++)
    {
        if (*(arr+i) == search_val)
        {
            return i;
        }
    }
    return -1;
}
```

array_search_binary.c:

```c
int search_binary(int* arr, int n, int search_val){
    int beg = 0;
    int end = n-1;
    while (beg <= end)
    {
        int mid = (beg + end) / 2;
        if (*(arr+mid) == search_val)
        {
            return mid;
        }
        else if (*(arr+mid) > search_val)
        {
            end = mid - 1;
        }
        else
        {
            beg = mid + 1;
        }
    }
    return -1;
}
```

Sorting:

array_sort_bubble.c:

```c
void sort_bubble(int *arr, int n) {
  for (int i = 0; i < n; i++)
  {
    for (int j = i + 1; j < n; j++)
    {
      if (*(arr+i) > *(arr+j))
      {
        int temp = *(arr+i);
        *(arr+i) = *(arr+j);
        *(arr+j) = temp;
      }
    }
  }
}
```

array_sort_selection.c:

```c
void sort_selection(int *arr, int n) {
  for (int i = n - 1; i >= 1; i--)
  {
    int max = *arr;
    int index = 0;
    for (int j = 1; j <= i; j++)
    {
      if (*(arr+j) > max)
      {
        max = *(arr+j);
        index = j;
      }
    }
    *(arr+index) = *(arr+i);
    *(arr+i) = max;
  }
}
```

array_sort_insertion.c:

```c
void sort_insertion(int *arr, int n)
{
    for (int i = 1; i < n; i++)
    {
        int temp = *(arr + i);
        for (int j = i - 1; j >= 0; j--)
        {
            if (*(arr + j) > temp)
            {
                *(arr + (j + 1)) = *(arr + j);
                *(arr + j) = temp;
            }
            else
            {
                break;
            }
        }
    }
}
```

array_sort_merge.c:

```c
#include <malloc.h>

int sizeOfOriginalArray;

void merge(int *, int, int, int);
void merge_sort(int *, int, int);


void merge(int* arr, int beg, int mid, int end)
{
    int i = beg, j = mid + 1, index = beg, k;
    int* temp = (int *)calloc(sizeOfOriginalArray, sizeof(int));
    while ((i <= mid) && (j <= end))
    {
        if (*(arr + i) < *(arr + j))
        {
            *(temp + index) = *(arr + i);
            i++;
        }
        else
        {
```

```
            *(temp + index) = *(arr + j);
            j++;
        }
        index++;
    }
    if (i > mid)

    {
        while (j <= end)
        {
            *(temp + index) = *(arr + j);
            j++;
            index++;
        }
    }
    else
    {
        while (i <= mid)
        {
            *(temp + index) = *(arr + i);
            i++;
            index++;
        }
    }
    for (k = beg; k < index; k++)
        *(arr + k) = *(temp + k);
}

void merge_sort(int* arr, int beg, int end)
{
    int mid;
    if (beg < end)
    {
        mid = (beg + end) / 2;
        merge_sort(arr, beg, mid);
        merge_sort(arr, mid + 1, end);
        merge(arr, beg, mid, end);
    }
}

void sort_merge(int *arr, int n) {
    sizeOfOriginalArray = n;
    // main merge sort function call
    merge_sort(arr, 0, n-1);
}
```

array_sort_quick.c:

```c
void quicksort(int*, int, int);
int partition(int*, int, int);

void quicksort(int *a, int beg, int end)
{
    int loc;
    if (beg < end)
    {
        loc = partition(a, beg, end);
        quicksort(a, beg, loc - 1);
        quicksort(a, loc + 1, end);
    }
}
int partition(int *a, int beg, int end)
{
    int left, right, loc, temp, flag = 0;
    left = loc = beg;
    right = end;
    while (flag != 1)
    {
        while (*(a + loc) <= *(a + right) && loc != right)
            right--;
        if (loc == right)
            flag = 1;
        else if (*(a + loc) > *(a + right))
        {
            temp = *(a + loc);
            *(a + loc) = *(a + right);
            *(a + right) = temp;
            loc = right;
        }
        if (flag != 1)
        {
            while (*(a + loc) >= *(a + left) && loc != left)
                left++;
            if (loc == left)
                flag = 1;
            else if (*(a + loc) < *(a + left))
            {
                temp = *(a + loc);
                *(a + loc) = *(a + left);
                *(a + left) = temp;
                loc = left;
            }
```

```
        }
    }
    return loc;
}

void sort_quick(int* arr, int n){
    quicksort(arr, 0, n - 1);
}
```

Other setup files:

array_insert_many.c:

```
void insert_many(int *arr, int *n)
{
    int noOfElements, val;
    printf("Enter the number of elements: ");
    scanf("%d", &noOfElements);

    for (int i = 0; i < noOfElements; i++)
    {

        printf("array[%d] = ", (*n));
        scanf("%d", &val);
        insert_at_last(arr, n, val);
    }
}
```

array_display_by_value.c:

```
void display_array(int arr[], int count){
    for (int i = 0; i < count; i++)
    {
        printf("%d", arr[i]);
        if (i < count - 1)
        {
            printf(", ");
        }

    }
}
```

array_case.c:

```c
void help()
{
    printf("0 - Help\n");
    printf("1 - Insert\n");
    printf("\t1 - Beginning\n");
    printf("\t2 - Middle\n");
    printf("\t3 - Last\n");
    printf("2 - Delete\n");
    printf("\t1 - Beginning\n");
    printf("\t2 - Middle\n");
    printf("\t3 - Last\n");
    printf("3 - Search\n");
    printf("\t1 - Linear\n");
    printf("\t2 - Binary\n");
    printf("4 - Sort\n");
    printf("\t1 - Bubble\n");
    printf("\t2 - Insertion\n");
    printf("\t3 - Selection\n");
    printf("\t4 - Merge\n");
    printf("\t5 - Quick\n");
    printf("5 - Display array\n");
    printf("6 - Insert many\n");
    printf("7 - Quit\n");
}

void insert(int *arr, int *n)
{
    int option, val, pos;
    printf("Enter the type of insertion: ");
    scanf("%d", &option);
    printf("Enter the value to be inserted: ");
    scanf("%d", &val);

    switch (option)
    {
    case 0:
        help();
        break;
    case 1:
        insert_at_beginning(arr, n, val);
        break;
    case 2:
        printf("Enter the position: ");
        scanf("%d", &pos);
```

```
        insert_at_position(arr, n, val, pos);
        break;
    case 3:
        insert_at_last(arr, n, val);
        break;

    default:
        printf("Enter a valid command\n");
        help();
        break;
    }
}

void delete (int *arr, int *n)
{
    int option, pos;
    printf("Enter the type of deletion: ");
    scanf("%d", &option);

    switch (option)
    {
    case 0:
        help();
        break;
    case 1:
        delete_from_beginning(arr, n);
        break;
    case 2:
        printf("Enter the position: ");
        scanf("%d", &pos);
        delete_from_position(arr, n, pos);
        break;
    case 3:
        delete_from_last(arr, n);
        break;

    default:
        printf("Enter a valid command\n");
        help();
        break;
    }
}

void search(int *arr, int n)
{
    int option, val, pos;
```

```
    printf("Enter the type of search: ");
    scanf("%d", &option);
    printf("Enter the value to be searched: ");
    scanf("%d", &val);

    switch (option)
    {
    case 0:
        help();
        break;
    case 1:
        pos = search_linear(arr, n, val);
        break;
    case 2:
        pos = search_binary(arr, n, val);
        break;

    default:
        printf("Enter a valid command\n");
        help();
        break;
    }

    if (pos > -1)
    {
        printf("a[%d] = %d\n", pos, val);
    }
    else
    {
        printf("Element not found.\n");
    }
}

void sort(int *arr, int n)
{
    int option;
    printf("Enter the type of sort: ");
    scanf("%d", &option);

    switch (option)
    {
    case 0:
        help();
        break;
    case 1:
        sort_bubble(arr, n);
```

```
        break;
    case 2:
        sort_insertion(arr, n);
        break;
    case 3:
        sort_selection(arr, n);
        break;
    case 4:
        sort_merge(arr, n);
        break;
    case 5:
        sort_quick(arr, n);
        break;

    default:
        printf("Enter a valid command\n");
        help();
        break;
    }
}
```

array_utilities.h:

```
#include "array_display_by_value.c"
#include "array_insert_beginning.c"
#include "array_insert_position.c"
#include "array_insert_last.c"
#include "array_insert_many.c"
#include "array_delete_beginning.c"
#include "array_delete_position.c"
#include "array_delete_last.c"
#include "array_search_linear.c"
#include "array_search_binary.c"
#include "array_sort_bubble.c"
#include "array_sort_selection.c"
#include "array_sort_insertion.c"
#include "array_sort_merge.c"
#include "array_sort_quick.c"
#include "array_case.c"
```

Main File:

main.c:

```c
#include <stdio.h>
#include "array_utilities.h"

int main()
{
    // number of elements initialized in array arr[]
    // must be equal to the value of variable n
    int arr[] = {};
    int n = 0;

    int option = 0;

    while (1)
    {
        switch (option)
        {
        case 0:
            help();
            break;
        case 1:
            insert(arr, &n);
            break;
        case 2:
            delete (arr, &n);
            break;
        case 3:
            search(arr, n);
            break;
        case 4:
            sort(arr, n);
            break;
        case 5:
            printf("array[%d] = {", n);
            display_array(arr, n);
            printf("}\n");
            break;
        case 6:
            insert_many(arr, &n);
            break;
        case 7:
            return 0;
```

```
            break;
        default:
            printf("Enter a valid command\n");
            help();
            break;
        }

        printf("> ");
        scanf("%d", &option);
    }

    return 0;
}
```

Output:

Insertion:

```
D:\codes\C\3rd-sem\sm-sir\array>gcc main.c && a.exe
0 - Help
1 - Insert
        1 - Beginning
        2 - Middle
        3 - Last
2 - Delete
        1 - Beginning
        2 - Middle
        3 - Last
3 - Search
        1 - Linear
        2 - Binary
4 - Sort
        1 - Bubble
        2 - Insertion
        3 - Selection
        4 - Merge
        5 - Quick
5 - Display array
6 - Insert many
7 - Quit
> 6
Enter the number of elements: 3
array[0] = 6
array[1] = 4
array[2] = 9
> 5
array[3] = {6, 4, 9}
> 1
Enter the type of insertion: 1
Enter the value to be inserted: 7
> 5
array[4] = {7, 6, 4, 9}
> 1
Enter the type of insertion: 2
Enter the value to be inserted: 8
Enter the position: 2
> 5
array[5] = {7, 6, 8, 4, 9}
> 1
Enter the type of insertion: 3
Enter the value to be inserted: 1
> 5
array[6] = {7, 6, 8, 4, 9, 1}
> 7

D:\codes\C\3rd-sem\sm-sir\array>
```

Deletion:

```
D:\codes\C\3rd-sem\sm-sir\array>gcc main.c && a.exe
0 - Help
1 - Insert
        1 - Beginning
        2 - Middle
        3 - Last
2 - Delete
        1 - Beginning
        2 - Middle
        3 - Last
3 - Search
        1 - Linear
        2 - Binary
4 - Sort
        1 - Bubble
        2 - Insertion
        3 - Selection
        4 - Merge
        5 - Quick
5 - Display array
6 - Insert many
7 - Quit
> 6
Enter the number of elements: 4
array[0] = 6
array[1] = 4
array[2] = 5
array[3] = 8
> 5
array[4] = {6, 4, 5, 8}
> 2
Enter the type of deletion: 1
> 5
array[3] = {4, 5, 8}
> 2
Enter the type of deletion: 2
Enter the position: 1
> 5
array[2] = {4, 8}
> 2
Enter the type of deletion: 3
> 5
array[1] = {4}
> 7

D:\codes\C\3rd-sem\sm-sir\array>
```

Sort:

```
D:\codes\C\3rd-sem\sm-sir\array>gcc main.c && a.exe
0 - Help
1 - Insert
        1 - Beginning
        2 - Middle
        3 - Last
2 - Delete
        1 - Beginning
        2 - Middle
        3 - Last
3 - Search
        1 - Linear
        2 - Binary
4 - Sort
        1 - Bubble
        2 - Insertion
        3 - Selection
        4 - Merge
        5 - Quick
5 - Display array
6 - Insert many
7 - Quit
> 6
Enter the number of elements: 5
array[0] = 5
array[1] = 8
array[2] = 1
array[3] = 4
array[4] = 6
> 5
array[5] = {5, 8, 1, 4, 6}
> 4
Enter the type of sort: 1
> 5
array[5] = {1, 4, 5, 6, 8}
> 7

D:\codes\C\3rd-sem\sm-sir\array>
```

Search:

```
D:\codes\C\3rd-sem\sm-sir\array>gcc main.c && a.exe
0 - Help
1 - Insert
        1 - Beginning
        2 - Middle
        3 - Last
2 - Delete
        1 - Beginning
        2 - Middle
        3 - Last
3 - Search
        1 - Linear
        2 - Binary
4 - Sort
        1 - Bubble
        2 - Insertion
        3 - Selection
        4 - Merge
        5 - Quick
5 - Display array
6 - Insert many
7 - Quit
> 6
Enter the number of elements: 5
array[0] = 1
array[1] = 5
array[2] = 4
array[3] = 3
array[4] = 8
> 5
array[5] = {1, 5, 4, 3, 8}
> 3
Enter the type of search: 1
Enter the value to be searched: 3
a[3] = 3
> 3
Enter the type of search: 1
Enter the value to be searched: 13
Element not found.
> 5
array[5] = {1, 5, 4, 3, 8}
> 4
Enter the type of sort: 5
> 5
array[5] = {1, 3, 4, 5, 8}
> 3
Enter the type of search: 2
Enter the value to be searched: 3
a[1] = 3
> 3
Enter the type of search: 2
Enter the value to be searched: 13
Element not found.
> 7

D:\codes\C\3rd-sem\sm-sir\array>
```

Repository:

https://github.com/Farhad618/bca-3rd-sem/tree/master/sm-sir/array

Thank You