

DS LAB 2

Date: 5/08/24

Name: Utkarsh Yadav

Roll: 23053172

2.1 WAP to create a 1-D array of n elements and perform the following menu based operations using function.

- a. insert a given element at specific position.
- b. delete an element from a specific position of the array.
- c. linear search to search an element
- d. traversal of the array

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void transverse(int* p, int size){
5     printf("The elements in the array are: \n");
6     for(int i=0; i<size; i++, p++){
7         printf("array[%d]: %d \n", i, *p);
8     }
9     free(p);
10 }
11
12 void insertElement(int *p, int size){
13
14     int element, index;
15     printf("Enter element to insert: \n");
16     scanf("%d", &element);
17
18     printf("Enter index to insert it at: \n");
19     scanf("%d", &index);
20
21     p = realloc(p, sizeof(int)*(size+1));
22     int dupSize = size;
23     for(int i = size+1; i>index; i--, dupSize--){
24         *(p+i) = *(p+dupSize);
25     }
26     *(p+index) = element;
27     printf("Updated Array: \n");
28     transverse(p, size+1);
29 }
30
31 void linearSearch(int* p, int size){
32     int element;
33     printf("Element to search for: ");
34     scanf("%d", &element);
35     for(int i=0; i<size; i++, p++){
36         if(*p == element){
37             printf("Element found at position %d", i+1);
38             free(p);
39             return;
40         }
41     }
42     printf("Element not found");
43     free(p);
44 }
45
46 void deleteElementbyIndex(int* p, int size){
47     int index;
48     printf("Index at which element is to be deleted: ");
49     scanf("%d", &index);
50
51     for(int i=size; i>index; index++){
52         *(p+index) = *(p+index+1);
53     }
54     printf("Updated array: \n");
55     transverse(p, size-1);
56 }
57
58
59 int main(){
60     int choice;
61     int size;
62     printf("Enter number of elements: ");
63     scanf("%d", &size);
64
65     int* array = (int*)malloc(sizeof(int)*size);
66
67     printf("Enter Elements: \n");
68     for(int i = 0; i<size; i++){
69         printf("array[%d] = ", i);
70         scanf("%d", &array[i]);
71     }
72
73     printf("***MENU***\n");
74     printf("1. INSERT\n");
75     printf("2. DELETE\n");
76     printf("3. LINEAR SEARCH\n");
77     printf("4. TRANSVERSE\n");
78     printf("5. EXIT\n");
79     printf("Enter a choice: ");
80     scanf("%d", &choice);
81
82     switch(choice){
83
84         case 1:
85             insertElement(array, size);
86             break;
87
88         case 2:
89             deleteElementbyIndex(array, size);
90             break;
91
92         case 3:
93             linearSearch(array, size);
94             break;
95
96         case 4:
97             transverse(array, size);
98             break;
99
100        case 5:
101            free(array);
102            printf("Exited the menu");
103            return 0;
104            break;
105
106    }
107 }
108
109
110 }

```

```
• $ ./menu
Enter number of elements: 5
Enter Elements:
array[0] = 1
array[1] = 2
array[2] = 3
array[3] = 4
array[4] = 5
***MENU***
1. INSERT
2. DELETE
3. LINEAR SEARCH
4. TRANSVERSE
5. EXIT
Enter a choice: 1
Enter element to insert:
50
Enter index to insert it at:
2
Updated Array:
The elements in the array are:
array[0]: 1
array[1]: 2
array[2]: 50
array[3]: 3
array[4]: 4
array[5]: 5
```

2.2 Write a program to perform the following operations on a given square matrix using functions:

- i. Find the no.of nonzero elements
- ii. Display upper triangular matrix
- iii. Display the elements of just above and below the main diagonal

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int** allocateMatrix(int rows, int cols);
5 void fillMatrix(int** array, int rows, int cols);
6 void printUpperTriangular(int** array, int rows, int cols);
7 int countNonZeroElements(int** array, int rows, int cols);
8 void printNonZeroAboveBelowDiagonal(int** array, int rows, int cols);
9 void freeMatrix(int** array, int rows);
10
11 int main() {
12     int rows, cols;
13
14     printf("Enter rows: ");
15     scanf("%d", &rows);
16     printf("Enter columns: ");
17     scanf("%d", &cols);
18
19     int** array = allocateMatrix(rows, cols);
20
21     fillMatrix(array, rows, cols);
22
23     printUpperTriangular(array, rows, cols);
24
25     int nonZeroCount = countNonZeroElements(array, rows, cols);
26     printf("Number of non-zero elements: %d\n", nonZeroCount);
27
28     printNonZeroAboveBelowDiagonal(array, rows, cols);
29     freeMatrix(array, rows);
30
31     return 0;
32 }
33
34 int** allocateMatrix(int rows, int cols) {
35     int** array = (int**)malloc(rows * sizeof(int*));
36     for (int i = 0; i < rows; i++) {
37         array[i] = (int*)malloc(cols * sizeof(int));
38     }
39     return array;
40 }
41
42 void fillMatrix(int** array, int rows, int cols) {
43     printf("Enter elements: \n");
44     for (int i = 0; i < rows; i++) {
45         for (int j = 0; j < cols; j++) {
46             printf("array[%d][%d] = ", i, j);
47             scanf("%d", &array[i][j]);
48         }
49     }
50 }
51
52 void printUpperTriangular(int** array, int rows, int cols) {
53     printf("Upper Triangular Matrix: \n");
54     for (int i = 0; i < rows; i++) {
55         for (int j = 0; j < cols; j++) {
56             if (j >= i) {
57                 printf("%d ", array[i][j]);
58             } else {
59                 printf(" ");
60             }
61         }
62         printf("\n");
63     }
64 }
65
66 int countNonZeroElements(int** array, int rows, int cols) {
67     int count = 0;
68     for (int i = 0; i < rows; i++) {
69         for (int j = 0; j < cols; j++) {
70             if (array[i][j] != 0) {
71                 count++;
72             }
73         }
74     }
75     return count;
76 }
77
78 void printNonZeroAboveBelowDiagonal(int** array, int rows, int cols) {
79     printf("Non-zero elements above the diagonal:\n");
80     for (int i = 0; i < rows; i++) {
81         for (int j = i + 1; j < cols; j++) {
82             if (array[i][j] != 0) {
83                 printf("array[%d][%d] = %d\n", i, j, array[i][j]);
84             }
85         }
86     }
87
88     printf("Non-zero elements below the diagonal:\n");
89     for (int i = 1; i < rows; i++) {
90         for (int j = 0; j < i; j++) {
91             if (array[i][j] != 0) {
92                 printf("array[%d][%d] = %d\n", i, j, array[i][j]);
93             }
94         }
95     }
96 }
97
98 void freeMatrix(int** array, int rows) {
99     for (int i = 0; i < rows; i++) {
100         free(array[i]);
101     }
102     free(array);
103 }
104
105

```

```
• $ ./sm
Enter rows: 3
Enter columns: 3
Enter elements:
array[0][0] = 1
array[0][1] = 2
array[0][2] = 3
array[1][0] = 4
array[1][1] = 5
array[1][2] = 6
array[2][0] = 7
array[2][1] = 8
array[2][2] = 9
Upper Triangular Matrix:
1 2 3
  5 6
    9
Number of non-zero elements: 9
Non-zero elements above the diagonal:
array[0][1] = 2
array[0][2] = 3
array[1][2] = 6
Non-zero elements below the diagonal:
array[1][0] = 4
array[2][0] = 7
array[2][1] = 8
```

2.3 WAP to represent a given sparse matrix in 3-tuple format using 2-D array.


```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(){
5     int rows, cols;
6
7     printf("Number of rows & columns: ");
8     scanf("%d %d", &rows, &cols);
9
10    int **array = (int **)malloc(rows * sizeof(int *));
11    for (int i = 0; i < rows; i++) {
12        array[i] = (int *)malloc(cols * sizeof(int));
13    }
14
15    printf("Enter the elements of the Sparse matrix:\n");
16    for (int i = 0; i < rows; i++) {
17        for (int j = 0; j < cols; j++) {
18            printf("Element [%d][%d]: ", i, j);
19            scanf("%d", &array[i][j]);
20        }
21    }
22
23    printf("Original Matrix: \n");
24
25    for (int i = 0; i < rows; i++) {
26        for (int j = 0; j < cols; j++) {
27            printf(" %d", array[i][j]);
28        }
29        printf("\n");
30    }
31
32
33    printf("3 tuple representation of Sparse Matrix: \n");
34
35    printf("\n r c v");
36
37    for(int i=0; i<rows; i++){
38        for(int j=0; j<cols; j++){
39            if(array[i][j] != 0){
40                printf("\n %d %d %d", i, j, array[i][j]);
41            }
42        }
43    }
44
45    for (int i = 0; i < rows; i++) {
46        free(array[i]);
47    }
48    free(array);
49
50    return 0;
51 }
```

```
Number of rows & columns: 3 3
Enter the elements of the Sparse matrix:
Element [0][0]: 1
Element [0][1]: 0
Element [0][2]: 0
Element [1][0]: 0
Element [1][1]: 2
Element [1][2]: 0
Element [2][0]: 0
Element [2][1]: 0
Element [2][2]: 3
Original Matrix:
1 0 0
0 2 0
0 0 3
3 tuple representation of Sparse Matrix:

r c v
0 0 1
1 1 2
2 2 3
```