**DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of Engineering)

Shahabad Daulatpur, Bawana Road, Delhi-110042



**DATA STRUCTURES LABORATORY**

**(SE-203n)**

**SUBMITTED TO:**

Mr. Ankur Narwal

Department of Software Engineering

**SUBMITTED BY:**

Aditya Kumar

23/SE/011

BTech SE 2nd year

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Objective** | **Date** | **Sign.** |
| 1. | To take input from the user an array of integers and reverse the array. | 21/08/24 |  |
| 2. | To perform various operations on a 1-Dimensional array using a menu driven program:   1. Insert an element into the array. 2. Delete an element into the array. 3. Finding the largest element in the array. 4. Finding the smallest element in the array. | 04/09/24 |  |
| 3. | To perform various string operations using a menu driven program:   1. Merging two strings. 2. Reversing a string. 3. Finding a substring in a given string and replacing it with another string. | 09/10/24 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**EXPERIMENT – 1**

**AIM:** To take input from the user an array of integers and reverse the array.

**CODE:**

#include <stdio.h>

void reverse(int arr[], int n) {

    for(int i = 0; i < n/2; i++) {

        int temp = arr[i];

        arr[i] = arr[n-1-i];

        arr[n-1-i] = temp;

    }

}

int main(void) {

    int n;

    printf("Enter The number of elements in the array: ");

    scanf("%d", &n);

    int arr[n];

    printf("Enter elements space seperated: ");

    for(int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);

    }

    reverse(arr, n);

    printf("Elements in array after reversal: \n");

    for(int i = 0; i < n; i++) {

        printf("%d ", arr[i]);

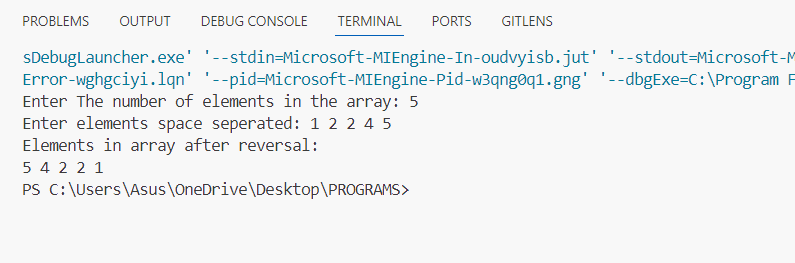
    }

    printf("\n");

    return 0;

}

**OUTPUT:**

****

**EXPERIMENT – 2**

**AIM:** To perform various operations on a 1-Dimensional array using a menu driven program:

1. Insert an element into the array.
2. Delete an element into the array.
3. Finding the largest element in the array.
4. Finding the smallest element in the array.

**CODE:**

#include <stdio.h>

#include <limits.h>

int largest(int\* arr, int n) {

    int maxi = INT\_MIN;

    for(int i = 0; i < n; i++) {

        if (maxi < arr[i]) maxi = arr[i];

    }

    return maxi;

}

int smallest(int\* arr, int n) {

    int mini = INT\_MAX;

    for(int i = 0; i < n; i++) {

        if (mini > arr[i] && arr[i] != -1) mini = arr[i];

    }

    return mini;

}

void insert(int \*arr, int n, int pos, int x) {

    int i = 0;

    while(i < pos) i++;

    int next = arr[i];

    arr[i] = x;

    int temp;

    while(i + 1 < n) {

        temp = arr[i+1];

        arr[i+1] = next;

        next = temp;

        i++;

    }

}

void delete\_from(int \*arr, int n, int pos) {

    int i = 0;

    while(i < pos) i++;

    int temp;

    while(i + 1 < n) {

        arr[i] = arr[i+1];

        i++;

    }

}

void display(int \*arr, int n) {

    printf("[ ");

    for(int i = 0; i < n; i++ ) {

        printf("%d ", arr[i]);

    }

    printf("]  "); printf("size = %d\n", n);

}

int main(void) {

    int y = 1;

    int choice;

    int n = 0;

    int MAX\_SIZE = 10;

    int arr[MAX\_SIZE];

    for(int i =0; i < MAX\_SIZE; i++) {

        arr[i] = -1;

    }

    printf("MENU DRIVER PROGRAM TO PERFORM OPERATIONS ON A ONE DIMENSIONAL ARRAY: \n");

    while(y) {

        printf("Choices: \n1. Initialise array. \n2. Insert elements into the array. \n3. Delete element from array.\n4. Find the smallest element in the array. \n5. Find the largest element in the array.\n");

        printf("choice: ");

        scanf("%d", &choice);

        switch(choice) {

            case 1:

                printf("Enter the number of elements :");

                scanf("%d", &n);

                printf("\nEnter elements space seperated :");

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

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

                }

                printf("An array of size %d has been initialised. \n", n);

                display(arr, n);

                break;

            case 2:

                int position\_insert;

                printf("Enter the position at which you want to insert element in the array :");

                scanf("%d", &position\_insert);

                if (position\_insert > MAX\_SIZE - 1 || position\_insert < 0) {printf("Invalid Position Passed. "); break;}

                printf("Enter the element that is to be inserted in the array :");

                int element;

                scanf("%d", &element);

                insert(arr, MAX\_SIZE, position\_insert, element);

                n = n+1;

                display(arr, n);

                printf("\n");

                break;

            case 3:

                int position\_delete;

                printf("Enter the position at which you want to delete element from the array :");

                scanf("%d", &position\_delete);

                if (position\_delete > MAX\_SIZE - 1 || position\_delete < 0) {printf("Invalid Position Passed. "); break;}

                delete\_from(arr, MAX\_SIZE, position\_delete);

                n = n-1;

                display(arr, n);

                printf("\n");

                break;

            case 5:

                printf("Maximum element in the array is: ");

                printf("%d\n", largest(arr, n));

                break;

            case 4:

                printf("Minimum element in the array is: ");

                printf("%d\n", smallest(arr, n));

                break;

            default:

                printf("Invalid choice inputted.");

                break;

        }

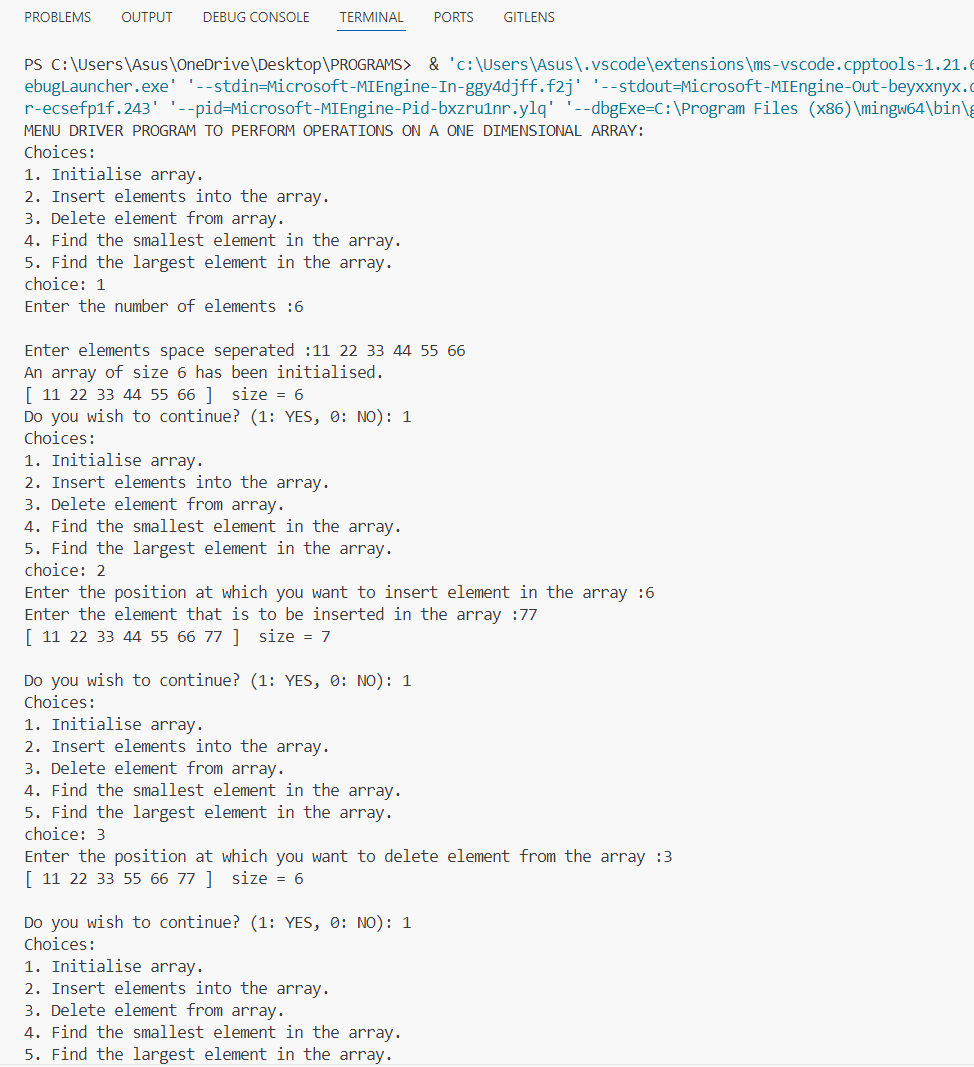
        printf("Do you wish to continue? (1: YES, 0: NO): ");

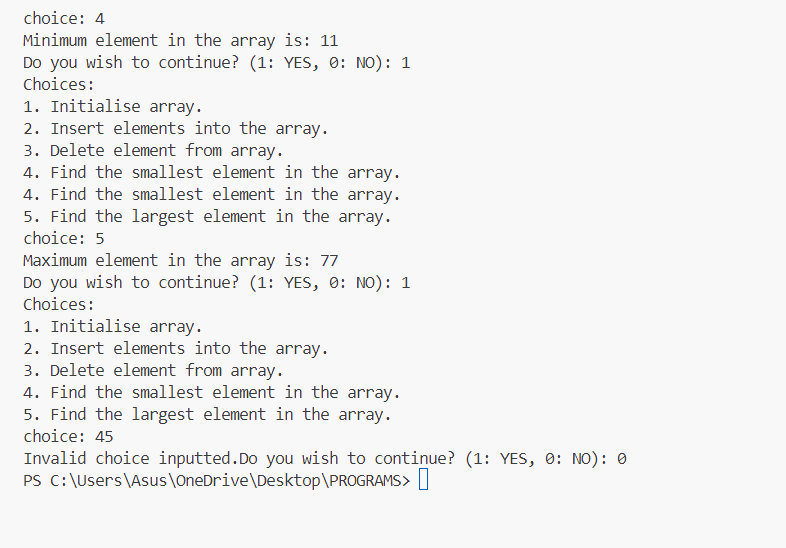
        scanf("%d", &y);

    }

}

**OUTPUT:**

****

****

**EXPERIMENT – 3**

**AIM:** To perform various string operations using a menu driven program:

1. Merging two strings.
2. Reversing a string.
3. Finding a substring in a given string and replacing it with another string.

**CODE:**

#include <stdio.h>

#include <string.h>

void merge\_strings() {

    char s1[100];

    char s2[100];

    printf("Enter the string-1: ");

    scanf("%s", s1);

    printf("Enter the string-2: ");

    scanf("%s", s2);

    printf("Merging string-2 into string-1..\n");

    int i = strlen(s1);

    int j;

    for(j = i; j<i+strlen(s2); j++) {

        s1[j] = s2[j-i];

    }

    s1[j] = '\0';

    printf("Resultant string is: %s", s1);

}

void reverse\_string() {

    char s1[100];

    printf("Enter the string that needs to be reversed: ");

    scanf("%s", s1);

    for(int i = 0; i < strlen(s1)/2; i++) {

        char temp = s1[i];

        s1[i] = s1[strlen(s1)-i-1];

        s1[strlen(s1)-i-1] = temp;

    }

    printf("Resultant string after reversal is: %s", s1);

}

void substringfindreplace(void) {

    char m[100]; // mainstring;

    printf("Enter the main string: ");

    scanf("%s", m);

    char find[100];

    printf("Enter the substring that needs to be replaced: ");

    scanf("%s", find);

    int  i= 0, j = 0;

    int start\_pos = -1, end\_pos = -1;

    for(int i = 0; i < strlen(m); i++) {

        if (m[i] == find[j]) {

            start\_pos = i;

            int dummy\_i = i;

            while(dummy\_i < strlen(m) && m[dummy\_i] == find[j]) {

                dummy\_i++; j++;

            }

            if (j == strlen(find))  {

                end\_pos = dummy\_i;

                break;

            }

            else j = 0;  start\_pos = -1;

        }

    }

    if (start\_pos == -1) {printf("Pattern/ substring not found in the given string."); return;}

    printf("Pattern found in main string \"%s\"  at position: %d", m, start\_pos);

    char replace[100];

    printf("\nEnter the string that should replace the pattern: ");

    scanf("%s", replace);

    char newstring[100];

    int k = 0;

    i = 0;

    while(i != start\_pos) {

        newstring[k] = m[i];

        k++; i++;

    }

    j = 0;

    while(j < strlen(replace)) {

        newstring[k] = replace[j];

        j++; k++;

    }

    i = end\_pos;

    while(m[i] != '\0') {

        newstring[k] = m[i];

        k++;i++;

    }

    newstring[k] = '\0';

    printf("New string after replacing the pattern: %s", newstring);

}

int main(void) {

    printf("PROGRAM-3: MENU DRIVEN PROGRAM FOR STRINGS.\n\n");

    char y = 'Y';

    do {

        printf("1. Merge two strings\n2. Reverse strings.\n3. Find a substring and replace it with another string.\n");

        int choice;

        printf("Enter choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                merge\_strings();

                break;

            case 2:

                reverse\_string();

                break;

            case 3:

                substringfindreplace();

                break;

            default:

                printf("Invalid choice entered.\n");

                break;

        }

        // Fix: Add a space before %c to consume the newline left by previous input

        printf("\n\nContinue? (Y/N): ");

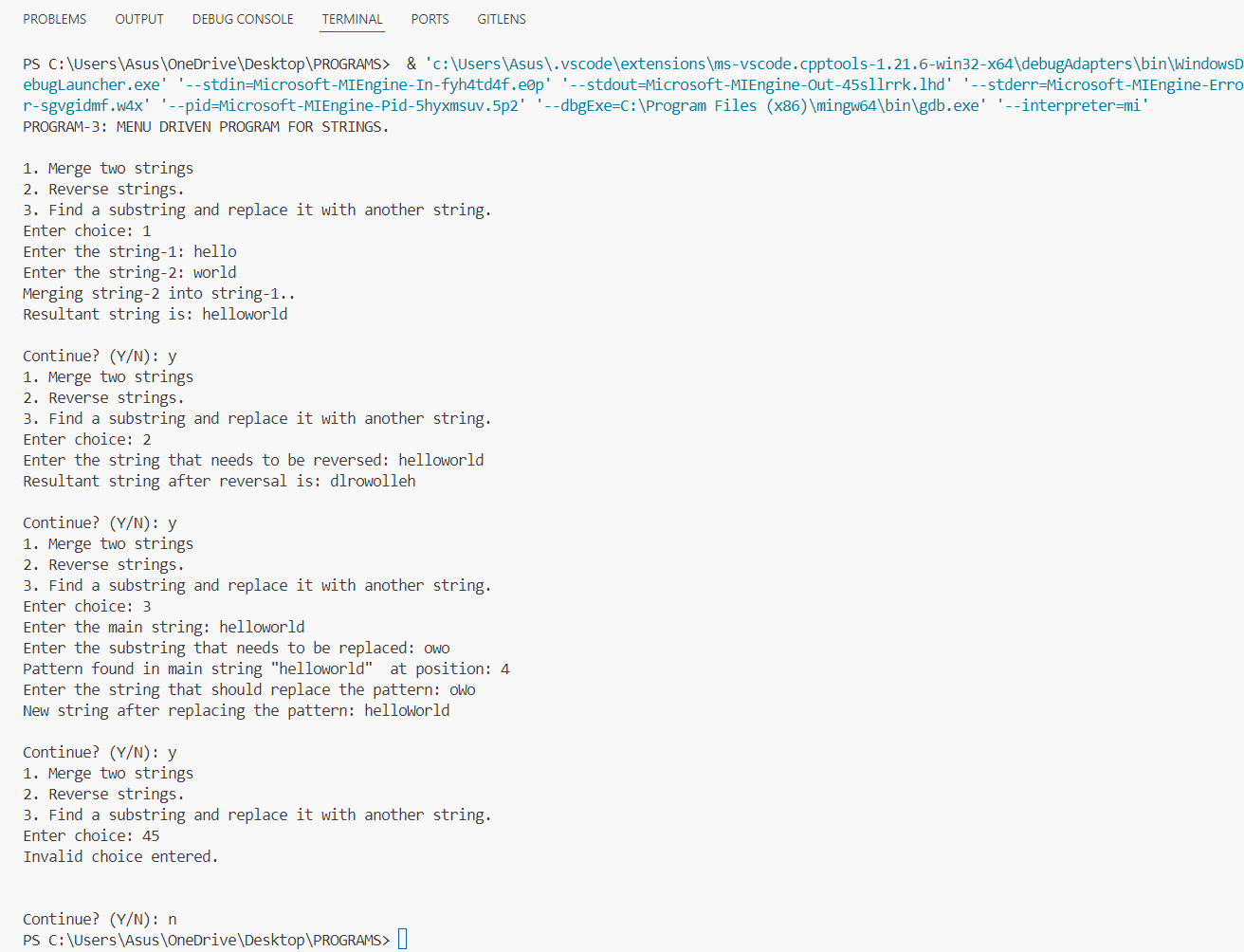
        scanf(" %c", &y);  // Notice the space before %c

    } while (y == 'Y' || y == 'y');  // Allow 'y' as well for case-insensitivity

    return 0;

}

**OUTPUT:**

****