**Data Structure**

**Lab-7**

**Submitted by: Submitted to:**

**Aakash Bhatt Pankaj Sir**

**500124633**

**(2023-2024)**

Exp no 1: WAP to demonstrate the union's effectiveness over structure. You can use any previously given structure program to depict the idea.

#include <stdio.h>

*// Define a structure for a rectangle*

struct *Rectangle*

{

    char shape\_type; *// 'R' for rectangle*

    float length;

    float width;

};

*// Define a structure for a circle*

struct *Circle*

{

    char shape\_type; *// 'C' for circle*

    float radius;

};

*// Define a union to store either a rectangle or a circle*

union *Shape*

{

    char shape\_type; *// To determine the type of shape ('R' for rectangle, 'C' for circle)*

    struct *Rectangle* rectangle;

    struct *Circle* circle;

};

int main()

{

    union *Shape* shape;

*// Store a rectangle in the union*

    shape.shape\_type = 'R';

    shape.rectangle.length = 5.0;

    shape.rectangle.width = 3.0;

    printf("Shape Type: %c\n", shape.shape\_type);

    printf("Rectangle Length: %.2f\n", shape.rectangle.length);

    printf("Rectangle Width: %.2f\n", shape.rectangle.width);

*// Store a circle in the same union*

    shape.shape\_type = 'C';

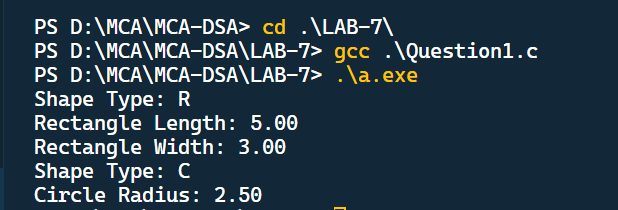
    shape.circle.radius = 2.5;

    printf("Shape Type: %c\n", shape.shape\_type);

    printf("Circle Radius: %.2f\n", shape.circle.radius);

    return 0;

}



Experiment no 2: WAP to demonstrate the various run-time memory allocation approaches like

 a. Malloc

 b. Calloc

 c. Free

 d. Realloc

For implementing this, make use of array, function, and wherever necessary pointer.

#include <stdio.h>

#include <stdlib.h>

*// Function to allocate memory using malloc*

int \*allocateWithMalloc(int size)

{

    int \*arr = (int \*)malloc(size \* sizeof(int));

    if (arr == NULL)

    {

        printf("Memory allocation with malloc failed.\n");

        exit(1);

    }

    return arr;

}

*// Function to allocate memory using calloc*

int \*allocateWithCalloc(int size)

{

    int \*arr = (int \*)calloc(size, sizeof(int));

    if (arr == NULL)

    {

        printf("Memory allocation with calloc failed.\n");

        exit(1);

    }

    return arr;

}

*// Function to reallocate memory using realloc*

int \*reallocate(int \*arr, int newSize)

{

    int \*newArr = (int \*)realloc(arr, newSize \* sizeof(int));

    if (newArr == NULL)

    {

        printf("Memory reallocation with realloc failed.\n");

        free(arr); *// Release the original memory*

        exit(1);

    }

    return newArr;

}

*// Function to print an array*

void printArray(int \*arr, int size)

{

    for (int i = 0; i < size; i++)

    {

        printf("%d ", arr[i]);

    }

    printf("\n");

}

int main()

{

    int \*dynamicArray = NULL;

    int size = 5;

*// Allocate memory using malloc*

    dynamicArray = allocateWithMalloc(size);

*// Initialize the array*

    for (int i = 0; i < size; i++)

    {

        dynamicArray[i] = i + 1;

    }

    printf("Array allocated with malloc: ");

    printArray(dynamicArray, size);

*// Reallocate memory using realloc*

    size = 10;

    dynamicArray = reallocate(dynamicArray, size);

*// Initialize the additional elements*

    for (int i = 5; i < size; i++)

    {

        dynamicArray[i] = i + 1;

    }

    printf("Array reallocated with realloc: ");

    printArray(dynamicArray, size);

*// Deallocate memory using free*

    free(dynamicArray);

    dynamicArray = NULL;

    return 0;

}

