Program1 swapping number using pointers

Code:-

#include<stdio.h>

void swap(int \*a,int \*b) {

int temp=\*a;

\*a=\*b;

\*b=temp;

}

int main(){

int num1,num2;

printf("enter the first number");

scanf("%d",&num1);

printf("enter the second number");

scanf("%d",&num2);

printf("Before swapping : num1=%d,num=%d\n",num1,num2);

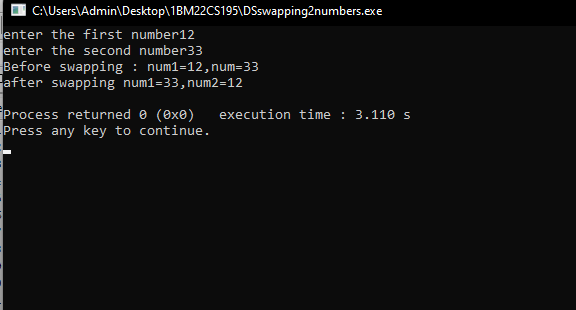
swap(&num1,&num2);

printf("after swapping num1=%d,num2=%d\n",num1,num2);

return 0;

}

Output:-



Program2 Dynamic memory allocation

Code:-

#include <stdio.h>

#include <stdlib.h>

void\* Malloc(size\_t size) {

return malloc(size);

}

void\* Realloc(void\* ptr, size\_t size) {

return realloc(ptr, size);

}

void\* Calloc(size\_t num, size\_t size) {

return calloc(num, size);

}

void Free(void\* ptr) {

free(ptr);

}

int main() {

int \*arr1, \*arr2;

size\_t size;

printf("Enter the size of the array: ");

scanf("%d", &size);

arr1 = (int\*)Malloc(size \* sizeof(int));

if (arr1 == NULL) {

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

return 0;

}

printf("Enter elements of the array:\n");

for (size\_t i = 0; i < size; i++) {

printf("Element %d: ", i + 1);

scanf("%d", &arr1[i]);

}

printf("Elements of the array (malloc):\n");

for (size\_t i = 0; i < size; i++) {

printf("%d ", arr1[i]);

}

printf("\n");

size \*= 2;

arr2 = (int\*)Realloc(arr1, size \* sizeof(int));

if (arr2 == NULL) {

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

Free(arr1);

return 1;

}

printf("Enter additional elements of the array:\n");

for (size\_t i = size / 2; i < size; i++) {

printf("Element %d: ", i + 1);

scanf("%d", &arr2[i]);

}

printf("Elements of the array (realloc):\n");

for (size\_t i = 0; i < size; i++) {

printf("%d ", arr2[i]);

}

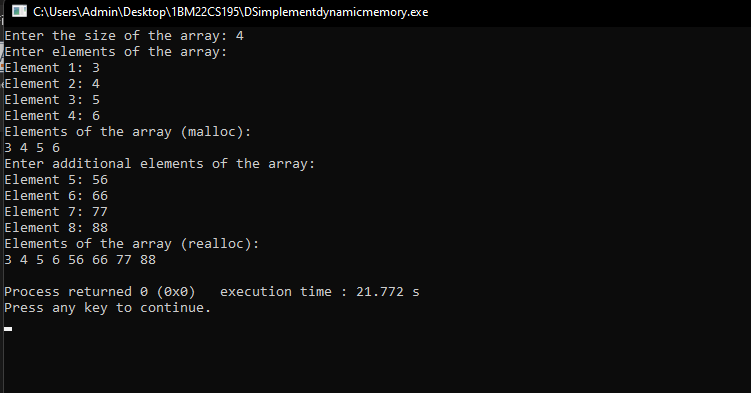
printf("\n");

Free(arr2);

return 0;

}

Output:-



Program3 Stack implementation

#include <stdio.h>

#include <stdlib.h>

#define MAX\_SIZE 10

struct Stack {

int items[MAX\_SIZE];

int top;

};

void initialize(struct Stack \*stack) {

stack->top = -1;

}

int isEmpty(struct Stack \*stack) {

return stack->top == -1;

}

int isFull(struct Stack \*stack) {

return stack->top == MAX\_SIZE - 1;

}

void push(struct Stack \*stack, int value) {

if (isFull(stack)) {

printf("Stack overflow. Cannot push %d.\n", value);

} else {

stack->top++;

stack->items[stack->top] = value;

printf("Pushed %d onto the stack.\n", value);

}

}

int pop(struct Stack \*stack) {

int poppedValue = -1;

if (isEmpty(stack)) {

printf("Stack underflow. Cannot pop from an empty stack.\n");

} else {

poppedValue = stack->items[stack->top];

stack->top--;

printf("Popped %d from the stack.\n", poppedValue);

}

return poppedValue;

}

void display(struct Stack \*stack) {

if (isEmpty(stack)) {

printf("Stack is empty.\n");

} else {

printf("Elements in the stack: ");

for (int i = 0; i <= stack->top; i++) {

printf("%d ", stack->items[i]);

}

printf("\n");

}

}

int main() {

struct Stack stack;

initialize(&stack);

push(&stack, 109);

push(&stack, 210);

push(&stack, 31);

push(&stack,40);

display(&stack);

pop(&stack);

display(&stack);

push(&stack, 405);

display(&stack);

return 0;

}

Output:-

