1. Swap

#include<stdio.h>

void swap(int \*x, int \*y){

int temp=\*x;

\*x=\*y;

\*y=temp;

}

int main(){

int x, y;

printf("Enter x and y\n");

scanf("%d%d",&x,&y);

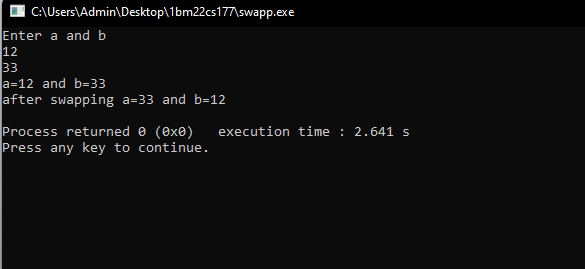
printf("x=%d and y=%d\n",x,y);

swap(&x,&y);

printf("after swapping x=%d and y=%d\n",x,y);

return 0;

}



1. Dynamic memory allocation

#include<stdio.h>

#include<stdlib.h>

int main(){

printf("Memory allocation through malloc\n");

int \*ptr=(int\*)malloc(5\*(sizeof(int)));

if(ptr==NULL){

printf("Memory allocation failed");

}

for(int i=0;i<5;i++){

ptr[i]=i;

printf("%d\t",ptr[i]);

}

printf("\nMemory allocation through calloc\n");

int \*ptr1=(int\*)calloc(4,(sizeof(int)));

if(ptr1==NULL){

printf("Memory allocation failed");

}

for(int i=0;i<4;i++){

ptr1[i]=i\*2;

printf("%d\t",ptr1[i]);}

printf("\nMemory allocation through realloc\n");

int \*ptr2=(int\*)realloc(ptr1,6\*(sizeof(int)));

if(ptr2==NULL){

printf("Memory re-allocation failed");

free(ptr1);

}

for(int i=4;i<6;i++){

ptr2[i]=i\*2;}

for(int i=0;i<6;i++){

printf("%d\t",ptr2[i]);}

free(ptr2);

printf("\nMemory after free\n");

for(int i=0;i<6;i++){

printf("%d\t",ptr2[i]);}

printf("\nMemory allocation through calloc\n");

ptr2=(int\*)calloc(4,(sizeof(int)));

if(ptr1==NULL){

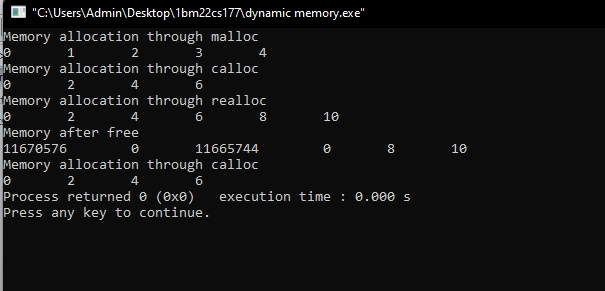
printf("Memory allocation failed");

}

for(int i=0;i<4;i++){

ptr2[i]=i\*2;

printf("%d\t",ptr2[i]);}

return 0;}

1. Stack operations

#include <stdio.h>

#include <stdlib.h>

#define max 100

int top = -1;

int stack[max];

void push(int a);

int pop();

void display();

int main() {

int arr[100], size;

printf("Enter array size: ");

scanf("%d", &size);

printf("Enter values of stack:\n");

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

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

push(arr[i]);

}

printf("Stack before popping:\n");

display();

for (int i = size - 1; i >= 0; i--) {

pop();

}

printf("Stack after popping:\n");

display();

return 0;

}

void push(int a) {

if (top == max - 1) {

printf("Stack overflow\n");

return;

}

top = top + 1;

stack[top] = a;

}

int pop() {

if (top == -1) {

printf("Stack underflow\n");

return -1;

}

top--;

return stack[top];

}

void display() {

if (top == -1) {

printf("Stack is empty\n");

return;

}

printf("Stack elements:\n");

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

printf("%d\t", stack[i]);

}

printf("\n");

}

