1. Write a program to print the address of a variable using pointer.

IPO

Input: variable say num and a pointer say ptr

Process: assigning the address of num to the pointer ptr

Output: address of the variable num

Code:

#include <stdio.h>

int main()

{

int num = 42;

int \*ptr = &num;

printf("Value of num: %d\n", num);

printf("Address of num: %p\n", (void\*)ptr);

return 0;

}

Output:



1. Write a program to access array elements using pointers.

IPO:

Input: array say arr and a pointer ptr

Process: using for loop printing the address of the elements in the variable

Output: access array elements using pointers

Code:

#include <stdio.h>

int main()

{

int arr[] = {10, 20, 30, 40, 50};

int \*ptr = arr;

printf("Array elements:\n");

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

{

printf("%d ", \*(ptr + i));

}

printf("\n");

return 0;

}

Output:



1. Write a program to swap two numbers using pointers.

IPO:

Input: two numbers say x and y

Process: swapping the address of the two variables

Ouput: swapping two numbers using pointers

Code:

#include <stdio.h>

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

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main()

{

int x = 5, y = 10;

printf("Before swap: x = %d, y = %d\n", x, y);

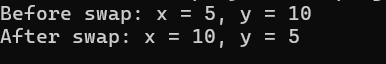
swap(&x, &y);

printf("After swap: x = %d, y = %d\n", x, y);

return 0;

}

Output:



1. Write a program to add two numbers using pointers.

IPO:

Input: two numbers say a and b

Process: store the addresses if the two numbers in pointers and and it

Output: Sum of two numbers using pointers

Code:

#include <stdio.h>

int main()

{

int a = 7, b = 3, sum;

int \*p1 = &a, \*p2 = &b;

sum = \*p1 + \*p2;

printf("Sum: %d\n", sum);

return 0;

}

Output:



1. Write a program to find the length of a string using pointers.

IPO:

Input: string say str

Process: using while loop incrementing the value of length for every character in the string using pointers

Output: length of the string using pointers

Code:

#include <stdio.h>

int main()

{

char str[] = "Hello";

char \*ptr = str;

int length = 0;

while (\*ptr != '\0') {

length++;

ptr++;

}

printf("Length of string: %d\n", length);

return 0;

}

Output:



1. Write a program to reverse a string using pointers.

IPO:

Input: string say str

Process: assigning the characters of str in another string in reverse order using pointers

Output: reverse a string using pointers

Code:

#include <stdio.h>

int main()

{

char str[] = "Pointer";

char \*start = str;

char \*end = str;

while (\*end != '\0')

{

end++;

}

end--;

char temp;

while (start < end)

{

temp = \*start;

\*start = \*end;

\*end = temp;

start++;

end--;

}

printf("Reversed string: %s\n", str);

return 0;

}

Output:



1. Write a program to count vowels using pointer.

IPO:

Input: string say str

Process: checking each character of the string using while loop and increnmenting the value of count if its a vowel

Ouput: count vowels using pointer

Code:

#include <stdio.h>

int isVowel(char ch)

{

if (ch == 'a' || ch == 'A' ||

ch == 'e' || ch == 'E' ||

ch == 'i' || ch == 'I' ||

ch == 'o' || ch == 'O' ||

ch == 'u' || ch == 'U') {

return 1;

}

return 0;

}

int main()

{

char str[] = "Programming in C";

char \*ptr = str;

int count = 0;

while (\*ptr != '\0')

{

if (isVowel(\*ptr))

{

count++;

}

ptr++;

}

printf("Number of vowels: %d\n", count);

return 0;

}

Output:



1. Write a program to demonstrate pointer to pointer.

IPO:

Input: variable say num

Process: assigning the address of the variable to ptr and again assigning ptr to pptr

Output: demonstrate pointer to pointer

Code:

#include <stdio.h>

int main()

{

int num = 50;

int \*ptr = &num;

int \*\*pptr = &ptr;

printf("Value of num: %d\n", num);

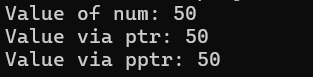
printf("Value via ptr: %d\n", \*ptr);

printf("Value via pptr: %d\n", \*\*pptr);

return 0;

}

Output:



1. Write a program to allocate memory using malloc() and free it.

IPO:

Input: variable n

Process: using malloc function and allocating memory , with the use of for loop.

Output:allocate memory using malloc() and free it.

Code:

#include <stdio.h>

#include <stdlib.h>

int main()

{

int \*arr;

int n = 5;

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

if (arr == NULL)

{

printf("Memory allocation failed\n");

return 1;

}

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

{

arr[i] = i + 1;

}

printf("Array elements: ");

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

{

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

}

printf("\n");

free(arr);

return 0;

}

Output:



1. Write a program to sort an array using pointer notation.

IPO:

Input: array arr

Process: using nested for loop and sorting the array by checking the pointer of each element with the other elements and switching their places if one pointer is greater than the other

Ouput: sort the array using pointers

Code:

#include <stdio.h>

void sort(int \*arr, int n)

{

for (int i = 0; i < n - 1; i++)

{

for (int j = i + 1; j < n; j++)

{

if (\*(arr + j) < \*(arr + i))

{

int temp = \*(arr + i);

\*(arr + i) = \*(arr + j);

\*(arr + j) = temp;

}

}

}

}

int main()

{

int arr[] = {5, 2, 9, 1, 3};

int n = 5;

sort(arr, n);

printf("Sorted array: ");

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

{

printf("%d ", \*(arr + i));

}

printf("\n");

return 0;

}

Output:

