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

IPO:

* Input: An integer variable
* Process: Use a pointer to store and print its address
* Output: Address of the variable

Program code:

#include <stdio.h>

Void main()

{

int a = 5;

int \*ptr = &a;

printf("Address of variable a: %p\n", ptr);

}

Sample output:

Address of variable a: 0x7ffeefbff4ac

2) Write a program to access array elements using pointers.

IPO:

* Input: An array of integers
* Process: Use a pointer to access and print each element
* Output: Array elements

Program code:

#include <stdio.h>

Void main()

{

int arr[5] = {1, 2, 3, 4, 5};

int \*ptr = arr;

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

{

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

}

}

Sample output:

Element 0: 1

Element 1: 2

Element 2: 3

Element 3: 4

Element 4: 5

3) Write a program to swap two numbers using pointers.

IPO:

* Input: Two integers
* Process: Swap values using pointer variables
* Output: Swapped values

Program code:

#include <stdio.h>

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

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main()

{

int x = 10, y = 20;

swap(&x, &y);

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

return 0;

}

Sample output:

After swap: x = 20, y = 10

4) Write a program to add two numbers using pointers.

IPO:

* Input: Two integers
* Process: Use pointers to add values
* Output: Sum of the two numbers

Program code:

#include <stdio.h>

int main()

{

int a = 7, b = 8;

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

int sum = \*p1 + \*p2;

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

return 0;

}

Sample output:

Sum = 15

5) Write a program to find the length of a string using pointers

IPO:

* Input: A string
* Process: Traverse the string using a pointer
* Output: Length of the string

Program code:

#include <stdio.h>

int main()

{

char str[] = "Pointer";

char \*ptr = str;

int length = 0;

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

length++;

ptr++;

}

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

return 0;

}

Sample output:

Length: 7

6) Write a program to reverse a string using pointers.

IPO:

* Input: A string
* Process: Use pointers to swap characters from start to end
* Output: Reversed string

Program code:

#include <stdio.h>

#include <string.h>

int main()

{

char str[] = "Pointer";

char \*start = str;

char \*end = str + strlen(str) - 1;

while (start < end)

{

char temp = \*start;

\*start = \*end;

\*end = temp;

start++;

end--;

}

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

return 0;

}

Sample output:

Reversed string: retnioP

7) Write a program to count vowels using pointer.

IPO:

* Input: A string
* Process: Traverse and count vowels using a pointer
* Output: Total number of vowels

Program code:

#include <stdio.h>

int main()

{

char str[] = "Pointer Example";

char \*ptr = str;

int count = 0;

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

{

char ch = \*ptr;

if (ch == 'a' || ch == 'e' || ch == 'i' ||

ch == 'o' || ch == 'u' ||

ch == 'A' || ch == 'E' || ch == 'I' ||

ch == 'O' || ch == 'U') {

count++;

}

ptr++;

}

printf("Vowel count: %d\n", count);

return 0;

}

Sample output:

Vowel count: 6

8) Write a program to demonstrate pointer to pointer.

IPO:

* Input: A single integer
* Process: Use pointer to pointer to access value
* Output: Value of the variable through double pointer

Program code:

#include <stdio.h>

int main()

{

int x = 50;

int \*p = &x;

int \*\*pp = &p;

printf("Value of x: %d\n", \*\*pp);

return 0;

}

Sample output:

Value of x: 50

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

IPO:

* Input: Size of the array
* Process: Allocate memory, assign values, print and free
* Output: Array elements

Program code:

#include <stdio.h>

#include <stdlib.h>

int main()

{

int n = 5;

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

if (arr == NULL)

{

printf("Memory not allocated.\n");

return 1;

}

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

arr[i] = i + 1;

}

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

{

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

}

free(arr);

return 0;

}

Sample output:

Element 0: 1

Element 1: 2

Element 2: 3

Element 3: 4

Element 4: 5

10) Write a program to sort an array using pointer notation.

IPO:

* Input: An array of integers
* Process: Use pointer notation in sorting algorithm
* Output: Sorted array

Program 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 + i) > \*(arr + j))

{

int temp = \*(arr + i);

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

\*(arr + j) = temp;

}

}

}

}

int main()

{

int arr[5] = {3, 1, 5, 2, 4};

sort(arr, 5);

printf("Sorted array: ");

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

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

}

printf("\n");

return 0;

}

Sample output:

Sorted array: 1 2 3 4 5