1. Print Address of a Variable using Pointer

IPO:

Input: None (uses a fixed integer variable)

Process: Store address of variable using pointer, print value and address

Output: Value and address of the variable

Code:

c

#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:

A screenshot of a computer

Description automatically generated

2. Access Array Elements using Pointers

IPO:

Input: Predefined array

Process: Use pointer arithmetic to access array elements

Output: Array elements printed using pointer

Code:

c

CopyEdit

#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:

A screenshot of a computer

Description automatically generated

3. Swap Two Numbers using Pointers

IPO:

Input: Two numbers

Process: Use pointer-based swap function

Output: Values of numbers after swapping

Code:

c

CopyEdit

#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:

A screenshot of a computer

Description automatically generated

4. Add Two Numbers using Pointers

IPO:

Input: Two numbers

Process: Add values using pointers

Output: Sum of the two numbers

Code:

c

CopyEdit

#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:

A screenshot of a computer

Description automatically generated

5. Find Length of a String using Pointers

IPO:

Input: String (predefined)

Process: Traverse string using pointer to count characters

Output: Length of the string

Code:

c

CopyEdit

#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:

A screenshot of a computer

Description automatically generated

6. Reverse a string using pointers

Input: String str = "Pointer"

Process: Swap characters from start and end using two pointers until they meet

Output: Reversed string: retnioP

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:

A screenshot of a computer

Description automatically generated

7. Count vowels using pointer

Input: String str = "Programming in C"

Process: Use pointer to traverse string and check each character for vowels

Output: Number of vowels: 5

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:

A screen shot of a computer

Description automatically generated

8. Demonstrate pointer to pointer

Input: Integer num = 50

Process: Access value through pointer ptr and pointer-to-pointer pptr

Output: Value of num: 50  
Value via ptr: 50  
Value via pptr: 50

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:

A screenshot of a computer

Description automatically generated

9. Allocate memory using malloc() and free it

Input: Integer n = 5

Process: Dynamically allocate memory using malloc, assign values, and free memory

Output: Array elements: 1 2 3 4 5

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:

A screenshot of a computer

Description automatically generated

10. Sort an array using pointer notation

Input: Array arr[] = {5, 2, 9, 1, 3}

Process: Sort array using nested loops and pointer notation \*(arr + i)

Output: Sorted array: 1 2 3 5 9

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:

A screenshot of a computer

Description automatically generated