1. Access addresses of different types of variables using pointers

In C or C++, you can use pointers to access the addresses of different types of variables. Here's how you can do it:

```
#include <stdio.h>

int main() {
    int intVar = 10;
    float floatVar = 5.5;
    char charVar = 'A';
    double doubleVar = 3.14159;

    // Declaring pointers for each type of variable
    int *intPtr;
    float *floatPtr;
    char *charPtr;
    double *doublePtr;

    // Assigning addresses to the pointers
    intPtr = &intVar;
    floatPtr = &floatVar;
    charPtr = &charVar;
    doublePtr = &doubleVar;

    // Displaying the addresses of the variables
    printf("Address of intVar: %p\n", intPtr);
    printf("Address of floatVar: %p\n", floatPtr);
    printf("Address of charVar: %p\n", charPtr);
    printf("Address of doubleVar: %p\n", doublePtr);

    return 0;
}
```

2. Swap two integers using pointers

To swap two integers using pointers, we need to use pointer dereferencing to access and modify the values at the addresses of the integers.

```
#include <stdio.h>

void swap(int *a, int *b) {
    int temp = *a;  // Dereference pointer 'a' to get
value
    *a = *b;        // Dereference pointer 'b' and
assign to 'a'
    *b = temp;      // Assign the temporary value to
'b'
}
int main() {
    int x = 5, y = 10;

    printf("Before swap: x = %d, y = %d\n", x, y);
    swap(&x, &y);  // Pass the addresses of x and y to
the swap function
    printf("After swap: x = %d, y = %d\n", x, y);

    return 0;
}
```

3. Compute area and perimeter of a rectangle using pointers as parameters to a function

We can compute the area and perimeter of a rectangle by passing the dimensions (length and width) as pointers to a function.

```
#include <stdio.h>

void computeRectangle(int *length, int *width, int
*area, int *perimeter) {
    *area = (*length) * (*width);          // Area =
length * width
    *perimeter = 2 * (*length + *width);   // Perimeter
= 2 * (length + width)
}
```

```
int main() {
    int length = 5, width = 3;
    int area, perimeter;

    computeRectangle(&length, &width, &area,
&perimeter);

    printf("Area of rectangle: %d\n", area);
    printf("Perimeter of rectangle: %d\n", perimeter);

    return 0;
}
```

4. Store values of an array and display it using pointers

In this example, we will store values in an array and display them using a pointer to traverse through the array.

```
#include <stdio.h>

int main() {
    int arr[] = {10, 20, 30, 40, 50};
    int *ptr = arr;  // Pointer to the first element of
the array

    // Display the array elements using the pointer
    printf("Array elements using pointer:\n");
    for (int i = 0; i < 5; i++) {
        printf("%d ", *(ptr + i));  // Dereference
pointer (ptr + i) to get array element
    }

    printf("\n");

    return 0;
}
```