Всичко е обяснено доста подробно на този линк: <https://www.geeksforgeeks.org/pointer-arithmetics-in-c-with-examples/>

Там се работи само с int\* и материата е сравнително лесна.

На кратко:

Тъй като всеки указател държи адрес, ние спокойно може да си прибавяваме числа към него, за да взимаме друг адрес на базата на указателя. Това действие се нарича аритметика с указатели.

{

int\* ptr = &var; //0x0d44

ptr+1 ------> това ще ни даде 0x0d48, без да променя указателя;

ptr += 1 ------> това е все едно да напишем (ptr = ptr +1), и така ще ни даде същата стойност от горе, но също така ще промени и на къде сочи указателя

}

Както се забелязва ptr+1 не ни дава директно следващия адрес. Това реално с колко адреса ще се отместим зависи от типа на указателя. Понеже int\* сочи към int, който е 4 байта, то като прибавим 1 ще се отиде на следващикя int, а 4рите байта "заети" от сегашния ще се прескочат. Тази методолога се използва за да може по ефикасно да итерираме през поредици елементи(масиви). Тък трябва да вметнем, че името на масива е реално просто указател към първия елемент. Именно това ни позволява да правим аритметиката без проблеми

int arr[3]; int\* ptr = a; ---> \*(a+2) е същото като а[2]и ще ни върне стойността на променливата еквивалента на втория индекс от масива.

За по-гъвкава работа с указатели може да се възползваме и от кастване(type casting), чието единствено предназаначение е да смени типа на дадения указател временно, без да променя адреса, към който сочи.

int\* ptr = &var; printf("%c", \*(char\*)ptr);

Тук дереферираме кастнатия към char\* указател. Следователно той ще се държи като char\* и при дерефериране ще ни върне само 1 байт. В случая първия байт от променливата вар.