Tutorial Link https://codequotient.com/tutorials/Pointer Arithmetic in C/5a006b53e63d6b7fd5dec2b3

TUTORIAL

Pointer Arithmetic in C

Chapter

Pointer arithmetic is very important in case of pointer management. As pointer is a special variable so operations performed with a pointer variable will behave in a special manner. We can do following operations on pointers: -

- Increment a pointer
- Decrement a pointer
- Add a constant to pointer
- Subtract a constant from pointer
- compare two pointers
- Subtract two pointers

Increment a pointer: if we increment 1 in a pointer variable then actually pointer will be incremented by the size of the variable it points to and not by 1 literally. So for example if an integer takes 4 bytes and we execute the following program it will increment by 4 not 1.

C

```c
#include <stdio.h>
int main ()
{
  int i=50;
  int *p=&i;
  printf("Value of p=%p \n", p);
  p++;
  printf("Value after p++=%p\n", p);
  return 0;
}

```

The output of the above program will be: -

```
Value of p = 0xbf952d38
Value after p++ = 0xbf952d3c
```

So the value after p++ is 4 ahead of the previous value. Similarly **pointer decrements** will work on the size of the variable not by the units.

Pointer arithmetic will be dangerous also in some situations, because with the help of these we can access any particular address of memory, which may not be intended for our program. And we might accidentally change some useful content of the memory without intending to do it. That's why the pointer arithmetic is important to use safely.

Following program will show various operations on pointers.

```c
#include <stdio.h>
int main()
{
  int number=50;
  int *p;        //pointer to int
  printf("Size of an integer variable is %ld locations\n", sizeof(int));
  printf("Address of variable number is %p \n", &number);
  p=&number;    //stores the address of number variable
  printf("Address of p variable is %p \n",p);
  p++;       //adding 1 to pointer variable
  printf("After adding 1: Address of p variable is %p \n",p);
  p=p+3;       //adding 3 to pointer variable
  printf("After adding 3: Address of p variable is %p \n",p);
  p--;           //subtracting 1 from pointer variable
  printf("After subtracting 1: Address of p variable is %p \n",p);
  p=p - 3;       //subtracting 3 from pointer variable
  printf("After subtracting 3: Address of p variable is %p \n",p);
  return 0;
}

```

The output of above program will be: -

```
Size of an integer variable is 4 locations
Address of variable number is 0xbfa70298
Address of p variable is 0xbfa70298
After adding 1: Address of p variable is 0xbfa7029c
After adding 3: Address of p variable is 0xbfa702a8
After subtracting 1: Address of p variable is 0xbfa702a4
After subtracting 3: Address of p variable is 0xbfa70298
```

In the above output, all the address may vary on your computer as address of memory will depend on hardware. And all the arithmetic is hexadecimal arithmetic which is base 16. So after adding 1 in pointer, it will be incremented by 4, and after adding 4 to the pointer, it will be incremented by (3*4)=12 so *0xbfa7029c + 12 = 0xbfa702a8 and so on.*

Video Solution

<iframe width="560" height="315" src="https://www.youtube.com/embed/Q3eboVxTuzg" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>