SNS COLLEGE OF TECHNOLOGY

Coimbatore-36.

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

COURSE NAME : 23CST101 PROBLEM SOLVING AND C PROGRAMMING

I YEAR/ V SEMESTER

UNIT – IV POINTERS

Operations on Pointers and Pointer Arithmatic

Department of Computer Science and Engineering

Pointer Arithmetic in C

We can perform arithmetic operations on the pointers like addition, subtraction, etc. However, as we know that pointer contains the address, the result of an arithmetic operation performed on the pointer will also be a pointer if the other operand is of type integer.

Following arithmetic operations are possible on the pointer in C

- Increment/Decrement of a Pointer
- Addition of integer to a pointer
- Subtraction of integer to a pointer
- Subtracting two pointers of the same type
- Comparison of pointers

Pointer Arithmetic in C

1. Increment/Decrement of a Pointer

Increment: It is a condition that also comes under addition. When a pointer is incremented, it actually increments by the number equal to the size of the data type for which it is a pointer.

For Example:

If an integer pointer that stores **address 1000** is incremented, then it will increment by 4(**size of an int**), and the new address will point to **1004**. While if a float type pointer is incremented then it will increment by 4(**size of a float**) and the new address will be **1004**.

2. Decrement: It is a condition that also comes under subtraction. When a pointer is decremented, it actually decrements by the number equal to the size of the data type for which it is a pointer.

For Example:

If an integer pointer that stores **address 1000** is decremented, then it will decrement by 4(**size of an int**), and the new address will point to **996**. While if a float type pointer is decremented then it will decrement by 4(**size of a float**) and the new address will be **996**

Pointer Arithmetic in C

Incrementing Pointer in C

If we increment a pointer by 1, the pointer will start pointing to the immediate next location. This is somewhat different from the general arithmetic since the value of the pointer will get increased by the size of the data type to which the pointer is pointing.

We can traverse an array by using the increment operation on a pointer which will keep pointing to every element of the array, perform some operation on that, and update itself in a loop.

The Rule to increment the pointer is given below:

```
new_address= current_address + i * size_of(data type)
```

Where i is the number by which the pointer get increased.

For 32-bit int variable, it will be incremented by 2 bytes.

For 64-bit int variable, it will be incremented by 4 bytes.

Pointer Arithmetic in C

Let's see the example of incrementing pointer variable on 64-bit architecture.

```
#include<stdio.h>
int main(){
int number=50;
int *p;//pointer to int
p=&number;//stores the address of number variable
printf("Address of p variable is %u \n",p);
p=p+1;
printf("After increment: Address of p variable is %u \n",p); // in our case, p will get incremented by 4 bytes.
return 0;
}
```

Output

```
Address of p variable is 3214864300
After increment: Address of p variable is 3214864304
```

Pointer Arithmetic in C

Traversing an array by using pointer

```c
#include<stdio.h>
void main ()
{
   int arr[5] = {1, 2, 3, 4, 5};
   int *p = arr;
   int i;
   printf("printing array elements...\n");
   for(i = 0; i< 5; i++)
   {
      printf("%d  ",*(p+i));
   }
}
```

Output

```
printing array elements...
1  2  3  4  5
```

Pointer Arithmetic in C

Decrementing Pointer in C

Like increment, we can decrement a pointer variable. If we decrement a pointer, it will start pointing to the previous location. The formula of decrementing the pointer is given below:

```
new_address= current_address - i * size_of(data type)
```

For 32-bit int variable, it will be decremented by 2 bytes.

For 64-bit int variable, it will be decremented by 4 bytes.

Pointer Arithmetic in C

Let's see the example of decrementing pointer variable on 64-bit OS.

```
#include <stdio.h>
void main(){
int number=50;
int *p;//pointer to int
p=&number;//stores the address of number variable
printf("Address of p variable is %u \n",p);
p=p-1;
printf("After decrement: Address of p variable is %u \n",p); // P will now point to the immidiate previous location.
}
```

Output

```
Address of p variable is 3214864300
After decrement: Address of p variable is 3214864296
```

Pointer Arithmetic in C

C Pointer Addition

We can add a value to the pointer variable. The formula of adding value to pointer is given below:

```
new_address= current_address + (number * size_of(data type))
```

32-bit

For 32-bit int variable, it will add 2 * number.

64-bit

For 64-bit int variable, it will add 4 * number.

Pointer Arithmetic in C

Let's see the example of adding value to pointer variable on 64-bit architecture.

```
#include<stdio.h>
int main(){
int number=50;
int *p;//pointer to int
p=&number;//stores the address of number variable
printf("Address of p variable is %u \n",p);
p=p+3;   //adding 3 to pointer variable
printf("After adding 3: Address of p variable is %u \n",p);
return 0;
}
```

Output

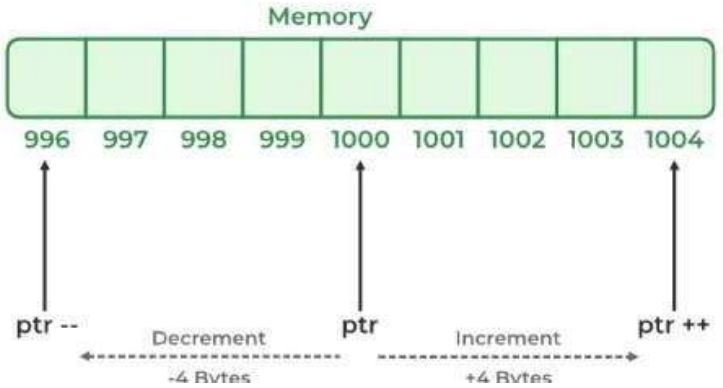
```
Address of p variable is 3214864300
After adding 3: Address of p variable is 3214864312
```

As you can see, the address of p is 3214864300. But after adding 3 with p variable, it is 3214864312, i.e., 4*3=12 increment.

```c
#include <stdio.h>
int main()
{
   int a = 22;
   int *p = &a;
   printf("p = %u\n", p); // p = 6422288
   p++;
   printf("p++ = %u\n", p); //p++ = 6422292     +4   // 4 bytes
   p--;
   printf("p-- = %u\n", p); //p-- = 6422288      -4   // restored to original value
    float b = 22.22;
   float *q = &b;
   printf("q = %u\n", q);  //q = 6422284
   q++;
   printf("q++ = %u\n", q); //q++ = 6422288       +4   // 4 bytes
   q--;
   printf("q-- = %u\n", q); //q-- = 6422284       -4  // restored to original value

   char c = 'a';
   char *r = &c;
   printf("r = %u\n", r);    //r = 6422283
   r++;
   printf("r++ = %u\n", r);    //r++ = 6422284     +1   // 1 byte
   r--;
   printf("r-- = %u\n", r);    //r-- = 6422283     -1  // restored to original value
    return 0;
}
```

Pointer Increment & Decrement



Output

```
p = 1441900792
p++ = 1441900796
p-- = 1441900792
q = 1441900796
q++ = 1441900800
q-- = 1441900796
r = 1441900791
r++ = 1441900792
r-- = 1441900791
```

Pointer Arithmetic in C

C Pointer Subtraction

Like pointer addition, we can subtract a value from the pointer variable. Subtracting any number from a pointer will give an address. The formula of subtracting value from the pointer variable is given below:

```
new_address= current_address - (number * size_of(data type))
```

32-bit

For 32-bit int variable, it will subtract 2 * number.

64-bit

For 64-bit int variable, it will subtract 4 * number.

Pointer Arithmetic in C

Let's see the example of subtracting value from the pointer variable on 64-bit architecture.

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

Output

```
Address of p variable is 3214864300
After subtracting 3: Address of p variable is 3214864288
```

You can see after subtracting 3 from the pointer variable, it is 12 (4*3) less than the previous address value.

Subtraction of Two Pointers

The subtraction of two pointers is possible only when they have the same data type.

Address2 - Address1 = (Subtraction of two addresses)/size of data type which pointer points

```c
#include<stdio.h>
void main ()
{
   int i = 100;
   int *p = &i;
   int *temp;
   temp = p;
   p = p + 3;
   printf("Pointer Subtraction: %d - %d = %d",p, temp, p-temp);
}
```

Output

```
Pointer Subtraction: 1030585080 - 1030585068 = 3
```


Thank
you