SNS COLLEGE OF TECHNOLOGY

Coimbatore-35.
An Autonomous Institution

COURSE NAME : 19CST101 PROGRAMMING FOR PROBLEM SOLVING

I YEAR/ I SEMESTER

UNIT-IV FUNCTIONS AND POINTERS

Topic: Pointers

Ms. Sumathi B
Assistant Professor
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
- Addition
- Subtraction
- Comparison

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.

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.

Pointers and Functions

In C programming, it is also possible to pass addresses as arguments to functions.

To accept these addresses in the function definition, we can use pointers. It's because pointers are used to store addresses. Let's take an example:

Example: Pass Addresses to Functions

```c
#include <stdio.h>
void swap(int *n1, int *n2);

int main()
{
    int num1 = 5, num2 = 10;

    // address of num1 and num2 is passed
    swap( &num1, &num2);

    printf("num1 = %d\n", num1);
    printf("num2 = %d", num2);
    return 0;
}

void swap(int* n1, int* n2)
{
    int temp;
    temp = *n1;
    *n1 = *n2;
    *n2 = temp;
}
```

When you run the program, the output will be:

```
num1 = 10
num2 = 5
```


Thank You!