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

Pointers and Arrays

Department of Computer Science and Engineering

UNIT IV

Pointers - Definition – Initialization –Operations on pointers-Pointer arithmetic –Pointers and arrays–Illustrative programs.

Pointers

Relationship Between Arrays and Pointers

An array is a block of sequential data. Let's write a program to print addresses of array elements.

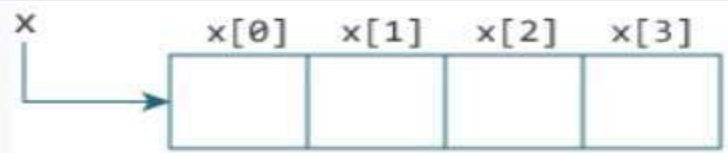
```c
#include <stdio.h>
int main() {
   int x[4];
   int i;

   for(i = 0; i < 4; ++i) {
      printf("&x[%d] = %p\n", i, &x[i]);
   }

   printf("Address of array x: %p", x);

   return 0;
}
```

Output

```
&x[0] = 1450734448
&x[1] = 1450734452
&x[2] = 1450734456
&x[3] = 1450734460
Address of array x: 1450734448
```

Pointers

Notice that, the address of `&x[0]` and `x` is the same. It's because the variable name `x` points to the first element of the array.



From the above example, it is clear that `&x[0]` is equivalent to `x`. And, `x[0]` is equivalent to `*x`.

Similarly,

- `&x[1]` is equivalent to `x+1` and `x[1]` is equivalent to `*(x+1)`.
- `&x[2]` is equivalent to `x+2` and `x[2]` is equivalent to `*(x+2)`.
- ...
- Basically, `&x[i]` is equivalent to `x+i` and `x[i]` is equivalent to `*(x+i)`.

Pointers

Example 1: Pointers and Arrays

```c
#include <stdio.h>
int main() {
  int i, x[6], sum = 0;
  printf("Enter 6 numbers: ");
  for(i = 0; i < 6; ++i) {
  // Equivalent to scanf("%d", &x[i]);
      scanf("%d", x+i);

  // Equivalent to sum += x[i]
      sum += *(x+i);
  }
  printf("Sum = %d", sum);
  return 0;
}
```

When you run the program, the output will be:

```
Enter 6 numbers:  2
 3
 4
 4
 12
 4
Sum = 29
```

Pointers

Example 2: Arrays and Pointers

```c
#include <stdio.h>
int main() {
  int x[5] = {1, 2, 3, 4, 5};
  int* ptr;

  // ptr is assigned the address of the third element
  ptr = &x[2];

  printf("*ptr = %d \n", *ptr);   // 3
  printf("*(ptr+1) = %d \n", *(ptr+1)); // 4
  printf("*(ptr-1) = %d", *(ptr-1));  // 2

  return 0;
}
```

Pointers

When you run the program, the output will be:

```
*ptr = 3
*(ptr+1) = 4
*(ptr-1) = 2
```

In this example, `&x[2]`, the address of the third element, is assigned to the `ptr` pointer. Hence, `3` was displayed when we printed `*ptr`.

And, printing `*(ptr+1)` gives us the fourth element. Similarly, printing `*(ptr-1)` gives us the second element.

Pointers

C Pointers

Pointers are powerful features of C and C++ programming. Before we learn pointers, let's learn about addresses in C programming.

Address in C

If you have a variable `var` in your program, `&var` will give you its address in the memory.

We have used address numerous times while using the `scanf()` function.

```
scanf("%d", &var);
```

Pointers

Here, the value entered by the user is stored in the address of `var` variable. Let's take a working example.

```c
#include <stdio.h>
int main()
{
  int var = 5;
  printf("var: %d\n", var);

  // Notice the use of & before var
  printf("address of var: %p", &var);
  return 0;
}
```

Output

```
var: 5
address of var: 2686778
```

Pointers

C Pointers

Pointers (pointer variables) are special variables that are used to store addresses rather than values.

Pointer Syntax

Here is how we can declare pointers.

```
int* p;
```

Here, we have declared a pointer `p` of `int` type.

You can also declare pointers in these ways.

```
int *p1;
int * p2;
```

Pointers

Let's take another example of declaring pointers.

```
int* p1, p2;
```

Here, we have declared a pointer `p1` and a normal variable `p2`.

Pointers

Assigning addresses to Pointers

Let's take an example.

```
int* pc, c;
c = 5;
pc = &c;
```

Here, 5 is assigned to the `c` variable. And, the address of `c` is assigned to the `pc` pointer.

Pointers

Get Value of Thing Pointed by Pointers

To get the value of the thing pointed by the pointers, we use the `*` operator. For example:

```
int* pc, c;
c = 5;
pc = &c;
printf("%d", *pc);    // Output: 5
```

Here, the address of `c` is assigned to the `pc` pointer. To get the value stored in that address, we used `*pc`.

Pointers

Note: In the above example, `pc` is a pointer, not `*pc`. You cannot and should not do something like `*pc = &c;`

By the way, `*` is called the dereference operator (when working with pointers). It operates on a pointer and gives the value stored in that pointer.

Pointers

Changing Value Pointed by Pointers

Let's take an example.

```
int* pc, c;
c = 5;
pc = &c;
c = 1;
printf("%d", c);    // Output: 1
printf("%d", *pc);  // Ouptut: 1
```

We have assigned the address of `c` to the `pc` pointer.

Then, we changed the value of `c` to 1. Since `pc` and the address of `c` is the same, `*pc` gives us 1.

Pointers

Let's take another example.

```
int* pc, c;
c = 5;
pc = &c;
*pc = 1;
printf("%d", *pc);  // Ouptut: 1
printf("%d", c);    // Output: 1
```

We have assigned the address of `c` to the `pc` pointer.

Then, we changed `*pc` to 1 using `*pc = 1;`. Since `pc` and the address of `c` is the same, `c` will be equal to 1.

Example: Working of Pointers

Let's take a working example.

```c
#include <stdio.h>
int main()
{
    int* pc, c;

    c = 22;
    printf("Address of c: %p\n", &c);
    printf("Value of c: %d\n\n", c);  // 22

    pc = &c;
    printf("Address of pointer pc: %p\n", pc);
    printf("Content of pointer pc: %d\n\n", *pc); // 22

    c = 11;
    printf("Address of pointer pc: %p\n", pc);
    printf("Content of pointer pc: %d\n\n", *pc); // 11

    *pc = 2;
    printf("Address of c: %p\n", &c);
    printf("Value of c: %d\n\n", c); // 2
    return 0;
}
```

Output

```
Address of c: 2686784
Value of c: 22

Address of pointer pc: 2686784
Content of pointer pc: 22

Address of pointer pc: 2686784
Content of pointer pc: 11

Address of c: 2686784
Value of c: 2
```


Thank
you