Unit VI – Part 03
(Pointers & Functions)

Lecture Slide

AS2023

Objectives

By the end of this session, students will be able to:

- Explain how pointers are used in functions
- How to pass addresses to Functions?
- How to pass pointers to Functions?

Pointers and Functions

- To accept these addresses in the function definition, we can use pointers.
- It's because pointers are used to store addresses.

Example: Pass Addresses to Functions

```
#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;
}
```

Output Sample:

num1 = 10
num2 = 5

Example: Pass Pointers to Functions

```
#include <stdio.h>
void addOne(int* ptr) {
        (*ptr)++; // adding 1 to *ptr
}
int main() {
        int* p, i = 10;
        p = &i;
        addOne(p);
        printf("*P = %d", *p); // 11 return 0;
}
```

Output Sample:

*P = 11

Passing Arguments to a function

- We pass two types of arguments to a function:
  1. Sending the values of the variable as the arguments referred to as ***call by value***
  2. Sending the address of the variable as the arguments referred to as ***call by reference***

Passing Arguments to a function

- ***Call-by-Value***
  - Here, the actual argument values in the calling function are copied into the corresponding formal arguments of the called function
  - Therefore, the effect of the modification of formal arguments is not visible to the actual arguments
- ***Call-by-Reference***
  - Here, the addresses of the actual arguments are sent to the formal arguments of the called function
  - Therefore, any modification of those argument values would be carried to the calling function as well

Call by Value vs Call by Reference

```
#Call by Value
#include <stdio.h>
void addOne(int a) {
        // adding 1 to ptr
        a++;
}
int main() {
        int p, i = 10;
        p = i;
        printf("Before=%d",p);
        addOne(p);
        printf("After=%d",p);
}
```

Output Sample:
Before = 10
After=10

```
#Call by Reference
#include <stdio.h>
void addOne(int* ptr) {
        // adding 1 to *ptr
        (*ptr)++;
}
int main() {
        int* p, i = 10;
        p = &i;
        printf("Before=%d",*p);
        addOne(p);
        printf("After=%d",*p);
}
```

Output Sample:
Before = 10
After=11

Pointer as function argument

- The addresses are used for manipulating the data
- When addresses are being passed to a function, the parameters receiving the address should be pointers
- Once change is made in the called function, this change will be also reflected in the original value of variables in calling function.
- This mechanism is also known as '*call by address*' or '*pass by pointers*'
- **DEMO:**WAP to compute the division of two numbers by using call by reference.
- **Homework:** WAP to solve quadratic equation using call by reference.

Thank you