SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

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

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

COURSE NAME : 23ITT101- PROBLEM SOLVING & C PROGRAMMING

I YEAR /I SEMESTER

Unit IV – FUNCTIONS AND POINTERS

Topic : Illustrative Programs

Example

```
1:  #include <stdio.h>
2:
3:  long cube(long x);/* Function prototype*/
4:
5:  long input, answer;
6:
7:  int main( void )  8:        {
9:  printf("Enter an integer value: ");
10:               scanf("%d", &input);
11: answer = cube(input);
12: printf("\nThe cube of %ld is %ld.\n", input,
answer);
13:
14: return 0;
15: }
16:
17: long cube(long  /* Function definition*/
x)
18: {
19:   long x_cubed;
20:
21:   x_cubed = x * x *
22:    x;  return
23: }x_cubed;
```

Arguments/formal parameter

Return data type

Actual parameters

/* calling function*/

- Function names is cube
- Variable that are requires is long
- The variable to be passed on is X(has single arguments)— value can be passed to function so it can perform the specific task. It is called arguments

Output

Enter an integer

value:4 The cube of 4

is 64.

A MULTI-FUNCTION PROGRAM

Example:

```
main()
{
printline();
printf("This illustrated the use of C functions \n");  printline();
}
printline()
{
int i;
for(i=1; i<40; i++)
pintf("-");
printf("\n");
}
```

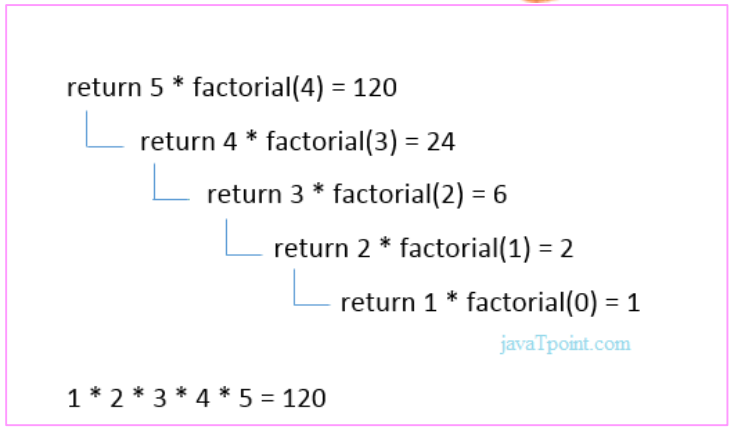
RETURN VALUES AND THEIR TYPES

```
/* C program to find SUM of two integer
Numbers using User Define Functions.*/

#include<stdio.h>

/*function declarations*/
int     sumint(int,int); /*to get sum*/
int main()
{
    int n1,n2;
    int sum;
    float avg;

    printf("Enter the first integer number: ");
    scanf("%d",&n1);

    printf("Enter the second integer number: ");
    scanf("%d",&n2);
```

```
     /*function calling*/
    sum=sumint(n1,n2);
  printf("Number1: %d, Number2: %d\n",n1,n2);
    printf("Sum: %d\n",sum);
     return 0;
}

/*function definitions*/
/* Function     : sumint
 * Arguments    : int,int - to pass two integer values
 * return type  : int - to return sum of values
*/
int sumint(int x,int y)
{
    /*x and y are the formal parameters*/
    int sum;
    sum=x+y;
    return sum;
}
```

Recursion

```
#include <stdio.h>

// Function prototype
int factorial(int n);
int main() {
    int num;
    printf("Enter a positive integer: ");
    scanf("%d", &num);

    if (num < 0) {
        printf("Factorial of a negative number is undefined.\n");
    } else {
        printf("The factorial of %d is %d\n", num, factorial(num));
    }
    return 0;
}
// Recursive function to calculate factorial
int factorial(int n) {
    if (n == 0 || n == 1) { // Base case: factorial(0) = 1 and factorial(1) = 1
        return 1;
    }
    return n * factorial(n - 1); // Recursive step
}
```



Pass arrays to a function in C

```
// Program to calculate the sum of array elements by passing to a function

#include <stdio.h>
float calculateSum(float num[]);

int main() {
  float result, num[] = {23.4, 55, 22.6, 3, 40.5, 18};

  // num array is passed to calculateSum()
  result = calculateSum(num);
  printf("Result = %.2f", result);
  return 0;
}
```

```
float calculateSum(float num[]) {
  float sum = 0.0;

  for (int i = 0; i < 6; ++i) {
    sum += num[i];
  }

  return sum;
}
```

Output:
Result = 162.50

Pass arrays to a function in C

Pass two-dimensional arrays

```
#include <stdio.h>
void displayNumbers(int num[2][2]);

int main() {
  int num[2][2];
  printf("Enter 4 numbers:\n");
  for (int i = 0; i < 2; ++i) {
    for (int j = 0; j < 2; ++j) {
      scanf("%d", &num[i][j]);
    }
  }

  // pass multi-dimensional array to a function
  displayNumbers(num);

  return 0;
}
```

```
void displayNumbers(int num[2][2]) {
  printf("Displaying:\n");
  for (int i = 0; i < 2; ++i) {
    for (int j = 0; j < 2; ++j) {
      printf("%d\n", num[i][j]);
    }
  }
}
```

Output:

Enter 4 numbers
2
3
4
5

Displaying:

2
3
4
5

Pass strings to a function in C

```
#include <stdio.h>
void displayString(char str[]);

int main()
{
    char str[50];
    printf("Enter string: ");
    fgets(str, sizeof(str), stdin);
    displayString(str);     // Passing string to a
function.
    return 0;
}
```

```
void displayString(char str[])
{
    printf("String Output: ");
    puts(str);
}
```

Understanding Pointers

```
#include<stdio.h>
void main( )
{
        char m;  int n;        float a, b;
        m = ‘A’;
        n = 150;
        a = 15.25, b = 10.75;
        printf(“%c is stored at addr %u.\n”, m, &m);
        printf(“%d is stored at addr %u.\n”, n, &n);
        printf(“%f is stored at addr %u.\n”, a, &a);
      printf(“%f is stored at addr %u.\n”, b, &b);
}
```

Increment and Decrement of a Pointer

```
#include <stdio.h>
// pointer increment and decrement
//pointers are incremented and decremented by the size of the data type they point to
int main()
{
    int a = 22;
    int *p = &a;
    printf("p = %u\n", p);     // p = 6422288
    p++;
    printf("p++ = %u\n", p);    //p++ = 6422292      // + 4 bytes
    p--;
    printf("p-- = %u\n", p);    //p-- = 6422288       // -4 bytes

    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;
}
```

Output:

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

Pointer Arithmetic on Arrays

```
// C program to illustrate the array traversal using pointers

#include <stdio.h>
int main()
{
    int N = 5;
    // An array
    int arr[] = { 1, 2, 3, 4, 5 };
    // Declare pointer variable
    int* ptr;
    // Point the pointer to first element in array arr[]
    ptr = arr;
    // Traverse array using ptr
    for (int i = 0; i < N; i++) {

        // Print element at which ptr points
        printf("%d ", ptr[0]);
        ptr++;
    }
}
```

```
Output:

1 2 3 4 5
```

Pointers and Arrays

Sample Program:

```
main ( )
{
   int  *p, sum, i;
   int x[5]  = {5, 9, 6, 3, 7};
   i  = 0;
   p  = x;          /* initializing with base address of x */
   printf("element   value    address \n\n");
   while (i <5)
   {
       printf("  x[%d]   %d   %u\n",  i , *p, p);
       sum  = sum  + *p;   /* accessing array element  */
       i++, p++;                /* incrementing pointer   */
   }
   printf("\n sum    =   %d\n", sum);
   printf("\n  &*[0]  =   %u\n",  &x[0];
   printf("\n  p      =   %u\n",   p);
}
```

Pointers and Arrays

```
#include <stdio.h>
int main()
{
int i;
int a[5] = {1, 2, 3, 4, 5};
int *p = a; // same as int*p = &a[0]
for (i = 0; i < 5; i++)
{
printf("%d", *p); p++;
}
return 0;
}
```

Pointers and Character Strings

```
main( )
 {
        char *name;
        int length;
        char *cptr=name;
        name=”delhi”;
        printf(“%s”,name);
        while (cptr != ‘\0’)
        {
                printf(“%c is stored at address %u \n”,*cptr,cptr);
                cptr++;
        }
        length=cptr-name;
        printf(“%d”,length);
}
```

Pointers and Character Strings

```
#include <stdio.h>
int main( )
{
char str[6] = "Hello"; // string variable
char *ptr = str; // pointer variable
while(*ptr != '\0') // print the string
{
printf("%c", *ptr);
// move the ptr pointer to the next memory location
ptr++;
}
return 0;
}
```

Array of Pointers

```
#include <stdio.h>
 int main()
{
      // Pointer to an array of five numbers
    int(*a)[5];
      int b[5] = { 1, 2, 3, 4, 5 };
      int i = 0;
      // Points to the whole array b
      a = &b;
      for (i = 0; i < 5; i++)
      printf("%d\n", *(*a + i));
      return 0;
}
```

Array of Pointers

```
// C program to demonstrate // example of array of pointers.
  #include <stdio.h>
  const int SIZE = 3;
  void main()
{
      // creating an array
    int arr[] = { 1, 2, 3 };
      // we can make an integer pointer array to
    // storing the address of array elements
    int i, *ptr[SIZE];
      for (i = 0; i < SIZE; i++) {
          // assigning the address of integer.
        ptr[i] = &arr[i];
    }
      // printing values using pointer
    for (i = 0; i < SIZE; i++) {
          printf("Value of arr[%d] = %d\n", i, *ptr[i]);
    }
}
```

Array of Pointers

```
#include<stdio.h>
  const int size = 4;
 void main()
{

    // array of pointers to a character
    // to store a list of strings
    char* names[] = {
        "amit",
        "amar",
        "ankit",
        "akhil"
    };
      int i = 0;
      for (i = 0; i < size; i++) {
        printf("%s\n", names[i]);
    }
}
```

Array of Pointers

// C program to understand difference between pointer to an integer and pointer to an array of integers.

```
#include<stdio.h>
int main()
{
// Pointer to an integer
int *p;
// Pointer to an array of 5 integers
int (*ptr)[5];
int arr[5];
// Points to 0th element of the arr.
p = arr;
// Points to the whole array arr.
ptr = &arr;
printf("p = %p, ptr = %p\n", p, ptr);
p++;
ptr++;
printf("p = %p, ptr = %p\n", p, ptr);
return 0;
}
```

Function Returning Pointers

```
#include <stdio.h>
int *returnPointer(int *p);
int main()
{
int i=10;
int *ptr1, *ptr2;
ptr1=&i;
ptr2=returnPointer(&i);
printf("\n *ptr1 = %d",*ptr1);
printf("\n *ptr2 = %d",*ptr2);
return 0;
}
int *returnPointer(int *pt)
{
return pt;   }
```

Function Returning Pointers

```
#include<stdio.h>
int *return_pointer(int *, int); // this function returns a pointer of type int
int main()
{
int i, *ptr;
int arr[] = {11, 22, 33, 44, 55};
i = 4;
printf("Address of arr = %p\n", arr);
ptr = return_pointer(arr, i);
printf("\nAfter incrementing arr by 4 \n\n");
printf("Address of ptr = %p\n\n", ptr);
printf("Value at %p is %d\n", ptr, *ptr);
// signal to operating system program ran fine
return 0;}
int *return_pointer(int *p, int n)
{
p = p + n;
return p;
}
```

Try it Yourself

What is the output of this C code?

```
int main()
{
int i = 10;
void *p = &i;
printf("%d\n", (int)*p);
return 0;
}
```

```
int main()
{
int i = 10;
void *p = &i;
printf("%f\n", *(float*)p);
return 0;
}
```

```
int main()
{
char *p = NULL;
char *q = 0;
if (p)
printf(" p ");
else
printf("nullp");
if (q)
printf("q\n");
else
printf(" nullq\n");
}
```


Thank
you