Lecture - 25

functions in C (Part 3)

Programming in C

```
#include <stdio.h>
# define PI 3.14
float arCircle ( float r ) {
    float area= PI * r * r ;
    return area;
}

int main( ) {
  float x= 10.0, res ;
  res = arCircle (x) ;
  printf("%f" , res);
  return 0;
}
```

macros

f^n name , return type = float

10.0 ~~r~~

f^n parameter (formal) $\rightarrow$ r is a float variable

$\text{ar} = \pi \times r^2$

area = 3.14 * 10.0 X 10.0

$\boxed{\text{area} = 314.000000}$

Return

entry point

declare

10.0 $\rightarrow$ arCircle (10.0)

$$\begin{tikzcd} x \arrow[d] \\ \text{arCircle (10.0)} \arrow[d] \\ r = 10.0 \end{tikzcd}$$

~~arCircle (x) ;~~ $\rightarrow$ 314.000000

function Call

Actual Parameter (Argument)

314.000000

input

res = arCircle (10.0)

$\Downarrow$

return

res $\Rightarrow$ 314.000000

$\Rightarrow$ $\boxed{\text{res} = 314.000000}$

function Call :

No value (empty Parameter)

```
Void greet ( ) {
    printf(" Good Morning");
}

int main ( ){
    greet () ──▷ Good Morning
    greet() ──▷ —— " ——
    greet() ──▷ —— " ——
    return 0;
}
```

* function are reusable, that means we can write the function ones & call it many times.

OOPs

* Encapsulation : Arranging the code within a particular block.

* Abstraction : Hiding the data

```c
void perimeter(float x, float y){
    float per = 2*(x+y);
    printf("The Perimeter of rectangle is %f", per);
}

int main(){
    float a=10, b=20;
 ① perimeter(a, b);
 ② perimeter(a+2, b+2);
    return 0;
}
```

a b

Copy of a and b

x y

a=10, b=20

12 22

Copies

Actual Parameter

There will be no change in the value of a & b

* Whenever we call a function, the actual Arguments provided during the function call are the copy of the values of actual source.

x और y का a और b की copy लिया है।

```c
void function (int x, int y) {
    x++; y++;
    printf("%d %d", x, y);
}

int main () {
    int x = 10, y = 20;
    function(y, x);
    printf("%d %d", x, y);
    return 0;
}
```

20 10

21 11

Inside function

x = 21 , y = 11

"21 11" ⟵ output

Outside f(x)

x = 10, y = 20

(call by values)

(copies

Positional Argument

function (20, 10)

y , x

arguments

* The variable defined / declared inside a function have only block level scope, whenever the f^n terminates then the variables also destroyed.

local

नष्ट

```
void simpleInterest ( int p, int t, float r = 0.10){
    float si = p * t * r ;
    printf("The simple interest is %f", si);
}

int main(){
    int x= 100000, y= 5;
    float z= 0.7;
    simpleInterest(x, y);
    simpleInterest(x, y, z);
    return 0;
}
```

Pre defined $\rightarrow$ default Argument

$\Downarrow$

The arguments that are defined during function definition inside the f^n parameters are called default argument

$$\begin{array}{ccc} p & t & r \\ \uparrow & \uparrow & \downarrow \\ 100000 & 5 & 0.10 \\ \uparrow & \uparrow & \end{array}$$

simpleInterest(x, y) $\rightarrow$ p= 100000, t= 5, r= 0.10

simpleInterest(x, y, z);

$$\begin{array}{ccc} 100000 & 5 & 0.7 \\ \downarrow & \downarrow & \downarrow \\ p & t & r \end{array}$$

$\rightarrow$ p= 100000

t= 5

r= 0.7 $\rightarrow$ Override the default argument

* If the value for default argument don not pass during function call it will that the default value defined for default argument.

Otherwise, it will surpass (override) the value for default argument and replace with new value passed.

* The default arguments are defined within rightmost position (location) in function parameter.

* When ever we call a function and pass the argument then the function takes the copies of passed values, that is called as 'Call by value'

$$\begin{tikzcd}
\text{Call by reference} \arrow[d, Rightarrow] & \\
\text{Global Scope} \arrow[d, Rightarrow] \arrow[r] & \boxed{\text{1 - 2 Lectures}} \arrow[d, Rightarrow] \\
\underline{\text{Basics of Pointers}} \arrow[d, Rightarrow] & \underline{\text{200 MCQ}} \\
\boxed{\text{Recursion}} &
\end{tikzcd}$$