In The Name Of God

Principles Of Programming

Session 3

Chapter 1

Functions – Call by reference & Call by Value

Presenter : Grader’s Team – Spring 2019

Functions and Parameters

- Syntax of function

Declaration :

```
<<return type>>  function name (parameter list);
```

Definition :

```
<<return type>> function name (parameter list) {
        body of the function;
}
```

Function Call :

```
Function name (parameter);
```

Example Function

```
#include <stdio.h>
void func (int x);          // declaration
void main(){
        func (10);          //Call
}
void func (int x){          //definition
       printf("%d", x);
}
```

Actual and Formal parameter

- Actual parameters are those that are used during a function call

- Formal parameters are those that are used in function definition and function declaration
- Global parameters are defined out of main and can be used in body of function

Arrays :

=> Type name [size] ;

The Integer Array :

```
int numbers [10] ; // the array would be filled with garbage
```

OR

```
int  numbers [ ] = { 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 }
```

OR

```
int len = 10 ;
int numbers [len] ;
```

The Character Arrays :

```
char sentence [10] ;
OR
char sentence  [6]  = { ‘a’ , ‘b’ , ‘c’ ,  ‘d’ , ‘\n’  , ‘\0’ }
 ‘\0’ is a character that has an ASCII code of zero .
OR
char sentence [ ] = { ‘a’ , ‘b’ , ‘c’ , ‘\0’ } ;
char sentence [ ] =  “abc” ;
char sentence [ ] = { “abc” } ;
// in the last two declerations , it puts ‘\0’ automatically
//Q: What is the difference between “a” and ‘a’ ?
```

Functions :

<return type > < function name > (< function argument > , ….) {

    // code

}

EX : adding 2 numbers and returning the result :

```
int adder ( int a , int b ) {
        int sum = a + b ;
        return sum ;
}
```

```
int adder ( int a , int b ) {
        return ( a + b ) ;
}
```

```
void print_result ( int a , int b ){
        int sum = a + b ;
        printf("%d" , sum );
}
```

Call By Reference & Value :

```
27
28  void changeValue(int a ){
29
30      a = a + 10 ; // AKA : a+= 10
31  }
32
33  void getName(char inputName[]){
34
35      int c = 0 ;
36      int i = 0 ;
37
38      while ( ( c = getchar()) != '\n'){
39          inputName[i++] = c ;
40      }
41      inputName[i] = '\0' ;
42  }
43
44
45  void printName(char name[]){
46
47      printf("Printing By Method I :\n");
48
49      int len = strlen(name);
50      int i = 0 ;
51      for ( i = 0 ; i < len ; i++){
52          putchar(name[i]);
53      }
54
55      putchar('\n');
56      //OR
57
58      printf("Printing By Method II \n");
59      i = 0 ;
60      while ( name[i] != '\0'){
61          putchar(name[i++]);
62      }
63  }
64
```

```c
#include <stdio.h>
#include <string.h>

void getName(char a[]);
void printName(char b[]);
void changeValue(int);
int main(void){

    //Variable Declarations :
    int a = 5 ;
    char name[100] ;


    // Call by Value :
    changeValue(a) ;
    printf("%d\n",a);


    //Call By References ( that's wrong anyway ) :
    printf("Enter Your Sentence : ") ;
    getName(name);
    printName(name);

    //End of The Program
    return 0 ;
}


```

Call by value

- Calling a function with parameters passed as values

```
int a=10;
func(a);
```

```
void func(int a){
        definition;
}
```

Here func(a) is call by value.

Any modiication done with in the function is local to it and it will not be effected outside the function.

Call by Reference

- Just for now , we accept that passing arrays to a function is known as Call by Reference . Just for now .

- We will show you an example in codes .

Example Call by value

```
#include <stdio.h>
void main(){
        int a=10;
        printf(“%d”, a);        // a=10;
        func (a);
        printf(“%d”, a);          //a=10;
}
void func (int x){
       printf(“%d”, x);             //a=10
       x++;
      printf(“%d”, x);             //a=11
}
```

Explanation