CS11001/CS11002 Programming and Data Structures (PDS)

(Theory: 3-1-0)

Functions without recursions

3 Steps to remember to use a function

1. Write the function.

2. Call the function (for now from main)

3. Declare the function

Declaration of a function

- int cube(int);
  - The first int denotes the return type of the function.
  - The second function is the type of the argument.
- This part should be mentioned early in the program(s).
- It informs the compiler that the function takes an integer as an input.

Functions with more than 1 arguments

- Write a function to find the maximum of 3 input numbers.
- int maxfunc(int i, int j, int k)

```
 {
   int max;
   if(i>=j  && i>=k) max = i;
   else if(j>=k) max = j;
   else max = k;
   return(max);
}
```

The main program

```
#include<stdio.h>
void main()
{
  int m. a, b, c;
  printf(“Input 3 numbers\n”);
  scanf(“%d %d %d”,&a,&b,&c);
  m=maxfunc(a,b,c);
  printf(“The max term is %d\n”,m);
}
```

Another way

```
#include<stdio.h>

main()
{
  int m, a, b, c;
  int max(int, int);
  printf("Enter 3 numbers:");
  scanf("%d %d %d",&a, &b, &c);
  printf("\na=%d, b=%d,
     c=%d\n",a,b,c);
  m=max(a,b);
  m=max(m,c);
  printf("Max term is %d\n",m);

}
```

```
int max(int a, int b)
{
 if(a>=b) return(a);
 else return(b);
}
```

Q: How will you use this to compute the max of n integers?

Where the Function is Declared?

- A function can be defined outside all functions. In that case they become global (just like variables declared outside)
  - All functions in the program, can then use these functions which are defined globally.
- Ex: #include<stdio.h>

```
      int cube(int);
      void main() {   }
       int cube(int n){   }
```

- Or, functions can be declared inside a function. In that case only that function where it is declared can use the declared function.

What is there to declare?

- In a C program, if the function comes after the calling function, the compiler needs to be informed via a declaration of:
  - The name of the function
  - The type of the value returned
  - The number and type of arguments that must be supplied in a call to the function.
- Note that if the function comes before the calling function then the declaration is not needed.
- Function declarations are also called function prototypes.

What is there to declare?

```
int cube(int n){
… /* Definition of the function*/
   }
/* The function is defined before being called. Hence
   the function declaration is not needed*/
void main(){
  int …
  cube(10);
 …
}
```

Return value of a function

- A function may or may not return value.
- If a function does not return anything it is a void function.
- In the function definition, the return type int is default and may be skipped. Though it is a good practice to explicitly state the return type.

The template for function definitions

```
Typename FunctionName(Paramterlist)
{
  statements declaring local variables;
  statements performing operations on
    variables;
  return(value);
/* return is a keyword and must be used. In
   case of a void function, this return step can
   be skipped*/
```

Passing parameters

- Function parameters are the means of communication between the calling and the called function.
- In C all parameters are passed **by value**.
  - Passing arguments by value means that the contents of the arguments in the calling function are not changed, even if they are changed in the called function.

Example

```
#include<stdio.h>
void myfunc(int num)
{
  printf(“In func the value of num is %d\n”,num);
  num=19;
 printf(“In func now value of num is %d\n”,num);
}
```

Example

```
void main()
{
  int num;
  num = 100;
  printf(“In main value of num is %d\n”,num);
  myfunc(num);
  printf(“After calling func, value of num is
   %d\n”,num);
}
```

An attempt to swap two integers

```
#include <stdio.h>
void swap(int a, int b)
{
  int temp;
  temp = a;
  a= b;
  b=temp;
}
void main()
{ scanf(“%d %d”,&a, &b); printf(“%d %d”,a,b);
  swap(a,b); printf(“%d %d”,a,b);
}
```

Both the printfs will print the same output! So, swapping does not take place. The variables in main are unupdated.

So, what may be done?

- In the function call swap, we also wanted to change values of variables, depending on the input from the user.
- While passing the variables to scanf, we precede the variables by an & symbol:
  - this meaning the address of the variable.
- So, instead of passing the two integers to the swap function, we pass the addresses of the integers we want to swap.

The swap program

```
void swap(int *a, int *b)
  { //*a denotes the value at the address
 int temp = *a;
*a = *b;
 *b = temp;
  }
 void main( )
 {int i, j; scanf("%d %d", &a, &b);
  printf("After swap: %d %d",a,b);
  swap(&a,&b);
  printf("After swap: %d %d",a,b);
}
```

Return values of functions

- constant: return(0);
- variable: return(a);
- user defined variables, general expressions
- Pointer to a function
- A function call (the call must return a value)