Technical Informatics I

Functions

by
Dr. Fatimah
Faculty of Mechanical Engineering
fatimahd@ump.edu.my

Functions

- Aims
  - Introduce students to functions and function prototypes
- Expected Outcomes
  - Students are able to construct user-defined functions in their C programs and appropriate function calls
- References
  - Harry H. Cheng, 2010. C for Engineers and Scientists: An Interpretive Approach, McGraw Hill

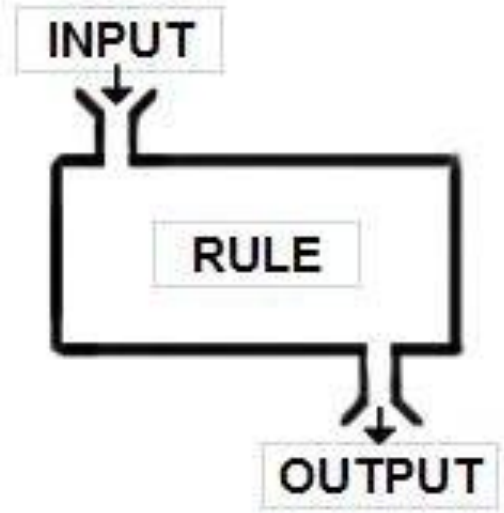
Content

- Introduction to functions
- Function Prototype
- User-defined function
  - Syntax
  - Variable declaration
  - Returning a value/not returning a value
- Examples:

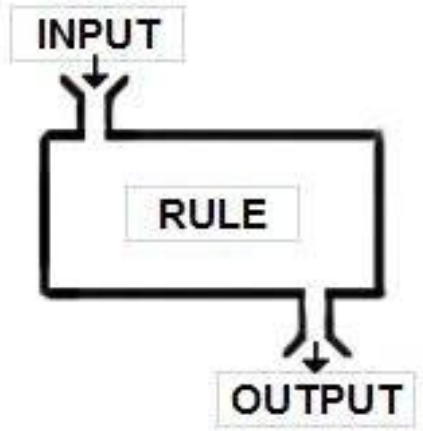
	Input Argument	Return Value
Example 2	1	✓
Example 3	×	✓
Example 4	1	×
Example 5	×	×
Example 6	Multiple arguments	×

Introduction to Functions

- A function is a named, independent section of C code that performs a specific task and optionally returns a value to the calling program or/and receives values(s) from the calling program.



Introduction: Functions



- You put something in, you may/may not get something out
  - Takes the input – applies some rule to it
  - And outputs whatever you have asked it to output
- It does the same thing every time

Why functions?

- Break longer jobs into conceptually smaller jobs which are precisely defined.
- A piece of a code which repeats at many places can be written only once and used again and again.
- Helps with debugging and maintenance

Example 1: Sequential Code

```c
/*Lab9ex1.c: Calculate Area of a circle*/

#include <stdio.h>
#include <math.h>
#define PI 3.14159265359

int main(){
  double A, r; /*declare Area, A, and r, radius*/

  printf("Enter value of radius, r (m): "); /*prompt user*/
  scanf("%lf",&r); /*Assign user input to cariable r*/

  /*calculate Area*/
  A = PI*pow(r,2); /*A=r^2*/

  printf("Area of a circle with radius, %.3f m is %.3f m^2\n",r,A);


  return 0;
}

```

```
Enter value of radius, r (m): 2
Area of a circle with radius, 2.000 m is 12.566 m^2
>Exit code: 0
```

Example 1: Sequential Code

```c
/*Lab9ex1.c: Calculate Area of a circle*/

#include <stdio.h>
#include <math.h>
#define PI 3.14159265359

int main(){
  double A, r; /*declare Area, A, and r, radius*/

  printf("Enter value of radius, r (m): "); /*prompt user*/
  scanf("%lf",&r); /*Assign user input to cariable r*/

  /*calculate Area*/
  A = PI*pow(r,2); /*A=r^2*/

  printf("Area of a circle with radius, %.3f m is %.3f m^2\n",r,A);


  return 0;
}
```

Main function

Function Prototype

- The function prototype declares the input and output parameters of the function. The function prototype has the following syntax:

```
return-type function-name(argument-list);
```

- Placed above the **`int main()`** function
- Function prototype includes information about the function's **`return-type`**, **`function-name`** and **`argument-list`**.

Function Prototype

```
1   /*Lab9ex1.c: Calculate Area of a circle*/
2
3   #include <stdio.h>
4   #include <math.h>
5   #define PI 3.14159265359
6
7
8
9
10
11
12  int main(){
13    double A, r; /*declare Area, A, and r, radius*/
14
15    printf("Enter value of radius, r (m): "); /*prompt user*/
16    scanf("%lf",&r); /*Assign user input to cariable r*/
17
18    /*calculate Area*/
19    A = PI*pow(r,2); /*A=r^2*/
20
21    printf("Area of a circle with radius, %.3f m is %.3f m^2\n",r,A);
22
23
24    return 0;
25  }
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
```

Function prototype

Main function

User defined function

a) Syntax

- The syntax of a function is

```
return-type function-name(argument-list) {
        Declarations and statements

}
```

- It is placed below the `int main()` function
- The statements (what you want the function to do) should be between the enclosed curly braces `{ }`
- When a function is called
  - The execution begins at the start of the function body
  - And terminates when it encounters the end of the closing brace } or encounters a `return`
  - After termination, the program returns to the calling function

User defined function

b) Local variables:

- **Local variables:** The variables declared within the body of the user-defined function
- These variables can *only* be used within the user-defined function (hence the term, *local* variable)
  - This implies, if you have a variable called A in your user-defined function, your main function will not recognise the variable.
- Local variables are used, declared and initialized like any other variables

User defined function

c) Returning a Value

- If function does not return a value
  - Then the function return type is type `void`.
  - Use `return` keyword only
  - Eg: `return;`
- If function returns a value
  - Use `return` keyword, followed by variable
  - Eg: `return area;`

- The value returned by `return` is passed back to the calling function
- The `return` value must match the return data type.

User defined function

```
1  /*Lab9ex1.c: Calculate Area of a circle*/
2
3  #include <stdio.h>
4  #include <math.h>
5  #define PI 3.14159265359
6
7
8
9
10
11
12 int main(){
13   double A, r; /*declare Area, A, and r, radius*/
14
15   printf("Enter value of radius, r (m): "); /*prompt user*/
16   scanf("%lf",&r); /*Assign user input to cariable r*/
17
18   /*calculate Area*/
19   A = PI*pow(r,2); /*A=r^2*/
20
21   printf("Area of a circle with radius, %.3f m is %.3f m^2\n",r,A);
22
23
24   return 0;
25 }
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
```

Function prototype

Main function

User defined function

Example 2: User defined function that takes in arguments and returns a value

- The syntax of a function is:

```
return-type function-name(argument-list)
   {
        Declarations and statements
   }
```

- `return-type`:
  - **`double`** – function returns the value of Area which is of type double
- `function-name`:
  - **`calculate_area`** – may give any name (Refer to GNU C Reference manual)
- `argument-list`:
  - **`double r`** – Argument input is radius (double)
- `return`:
  - returns Area

Example 2: User defined function that takes in arguments and returns a value

- The syntax of the function is:

```
double calculate_area(double r)
    {
        double A;
       A = PI*pow(r,2);
       return A;
    }
```

Example 2: User defined function that takes in arguments and returns a value

- The function prototype has the following syntax:

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

```
double calculate_area(double r)
    {
        double A;
       A = PI*pow(r,2);
       return A;
    }
```

Example 2: User defined function that takes in arguments and returns a value

- Function prototype:

```
double calculate_area(double r);
```

Example 2: User defined function that takes in arguments and returns a value

```c
/*Lab9ex2.c: Calculate Area of a circle*/
#include <stdio.h>
#include <math.h>
#define PI 3.14159265359


double calculate_area(double r);


int main(){
  double A, r; /*declare Area, A, and r, radius*/

  printf("Enter value of radius, r (m): "); /*prompt user*/
  scanf("%lf",&r); /*Assign user input to cariable r*/

  /*calculate Area*/
  A = calculate_area(r); /*A=r^2*

  printf("Area of a circle with radius, %.3f m is %.3f m^2\n",r,A);


  return 0;
}

double calculate_area(double r)
{
    double A;
    A = PI*pow(r,2); /*A=r^2*/
    return A;
}
```

Function prototype

Main function

Calling function

User defined function

Example 3: User defined function that does not takes in arguments and returns a value

- The syntax of a function is:

```
return-type function-name(argument-list)
    {
        Declarations and statements
    }
```

- **return-type**:
  - **double** - function returns the value of Area which is of type double
- **function-name**:
  - **calculate_area** - may give any name (Refer to GNU C Reference manual)
- **argument-list**:
  - **void** - No argument input. But since calculation requires r, need to place `scanf` statement inside this function
- **return**:
  - returns Area

Example 3: User defined function that does not takes in arguments and returns a value

- The syntax of the function is:

```
double calculate_area(void)
{
    double A, r;
    printf("Enter value of radius, r (m): ");
    scanf("%lf",&r);
    A = PI*pow(r,2); /*A=r^2*/
    return A;
}
```

Example 3: User defined function that does not takes in arguments and returns a value

- The syntax of the function prototype is:

```
double calculate_area(void)
{
    double A, r;
    printf("Enter value of radius, r (m): ");
    scanf("%lf",&r);
    A = PI*pow(r,2); /*A=r^2*/
    return A;
}
```

Example 3: User defined function that does not takes in arguments and returns a value

- Function prototype:

```
double calculate_area(void);
```

Example 3: User defined function that does not takes in arguments and returns a value

```
1  /*Lecture 8: Functions*/
2  /*Example 3: Calculate Area of a circle*/
3  #include <stdio.h>
4  #include <math.h>
5  #define PI 3.14159265359
6
7
8  double calculate_area(void);
9
10
11 int main(){
12   double A; /*declare Area, A, and r, radius*/
13
14   /*calculate Area*/
15   A = calculate_area(); /*A=r^2*/
16
17   printf("Area of a circle is %.3f m^2\n",A);
18   return 0;
19 }
20
21 double calculate_area(void)
22 {
23     double A, r;
24     printf("Enter value of radius, r (m): "); /*prompt user*/
25     scanf("%lf",&r); /*Assign user input to cariable r*/
26     A = PI*pow(r,2); /*A=r^2*/
27     return A;
28 }
```

Function prototype

Main function

Calling function

User defined function

Example 4: User defined function that takes in arguments and does not returns a value

- The syntax of a function is:

```
return-type function-name(argument-list)
    {
        Declarations and statements
    }
```

- **return-type**:
  - **void** - does not return a value
- **function-name**:
  - **calculate_area** – may give any name (Refer to GNU C Reference manual)
- **argument-list**:
  - **double r** - Argument input is radius (double)
- **return**:
  - **return** - function does not return any value

Example 4: User defined function that takes in arguments and does not returns a value

- The syntax of the function is:

```
void calculate_area(double r)
{
    double A;
    A = PI*pow(r,2); /*A=r^2*/
    printf("Area of a circle with radius, %.3f m is %.3f
  m^2\n",A,r);
    return;
}
```

Example 4: User defined function that takes in arguments and does not returns a value

- The syntax of the function prototype is:

```
void calculate_area(double r)
{
    double A;
    A = PI*pow(r,2); /*A=r^2*/
    printf("Area of a circle with radius, %.3f m is %.3f
  m^2\n",A,r);
    return;
}
```

Example 4: User defined function that takes in arguments and does not returns a value

- Function prototype:

```
void calculate_area(double r);
```

Example 4: User defined function that takes in arguments and does not returns a value

```c
/*lab9ex6.c: Calculate Area of a circle*/
#include <stdio.h>
#include <math.h>
#define PI 3.14159265359


void calculate_area(double r);

int main(){
  double A, r;
  printf("Enter value of radius, r (m): "); /*prompt user*/
  scanf("%lf",&r); /*Assign user input to cariable r*/

  /*calculate Area*/
  calculate_area(r); /*A=r^2*/

  return 0;
}

void calculate_area(double r)
{
    double A;
    A = PI*pow(r,2); /*A=r^2*/
    printf("Area of a circle with radius, %.3f m is %.3f m^2\n",r,A);
    return;
}
```

Function prototype

Main function

Calling function

User defined function

Example 5: User defined function that does not take in arguments and does not returns a value

The syntax of a function is:

```
return-type function-name(argument-list)
   {
        Declarations and statements
   }
```

- **return-type**:
  - **void** - does not return a value
- **function-name**:
  - **calculate_area** – may give any name (Refer to GNU C Reference manual)
- **argument-list**:
  - **Void** - No argument input. But since calculation requires r, need to place **scanf** statement inside this function
- **return**:
  - **return** - function does not return any value

Example 5: User defined function that does not take in arguments and does not returns a value

- The syntax of the function is:

```
void calculate_area(void)
{
    double A;
    printf("Enter value of radius, r (m): ");
    scanf("%lf",&r);
    A = PI*pow(r,2);
    printf("Area of a circle with radius, %.3f m is %.3f m^2\n",A,r);
    return;
}
```

Example 5: User defined function that does not take in arguments and does not returns a value

- The syntax of the function prototype is:

```
void calculate_area(void)
{
    double A, r;
    printf("Enter value of radius, r (m): ");
    scanf("%lf",&r);
    A = PI*pow(r,2);
    printf("Area of a circle with radius, %.3f m is %.3f m^2\n",A,r);
    return;
}
```

Example 5: User defined function that does not take in arguments and does not returns a value

- Function prototype:

```
void calculate_area(void);
```

Example 5: User defined function that does not take in arguments and does not returns a value

```
1   /*Lecture 8: Functions*/
2   /*Example 5: Calculate Area of a circle*/
3   #include <stdio.h>
4   #include <math.h>
5   #define PI 3.14159265359
6
7
8   void calculate_area(void);
9
10
11  int main(){
12
13    /*calculate Area*/
14    calculate_area(); /*A=r^2*/
15
16    return 0;
17  }
18
19  void calculate_area(void)
20  {
21      double A, r; /*declare Area, A, and r, radius*/
22      printf("Enter value of radius, r (m): "); /*prompt user*/
23      scanf("%lf",&r); /*Assign user input to cariable r*/
24      A = PI*pow(r,2); /*A=r^2*/
25      printf("Area of a circle with radius, %.3f m is %.3f m^2\n",r,A);
26      return;
27  }
28
```

Function prototype

Calling function

Main function

User defined function

Example 6: User defined function that does takes in multiple arguments

The syntax of a function is:

```
return-type function-name(argument-list)
    {
        Declarations and statements
    }
```

- `return-type`:
  - **`void`** - does not return a value
- `function-name`:
  - **`calculate_area`** – may give any name (Refer to GNU C Reference manual)
- `argument-list`:
  - **`double r, double A`** – For an example of more than 1 argument(multiple arguments)
- `return`:
  - **`return`** - function does not return any value

Example 6: User defined function that does takes in multiple arguments

- The syntax of the function is:

```
void calculate_area(double A, double r)
{
     printf("Area of a circle with radius, %.3f m is %.3f
   m^2\n",A,r);
    return;
}
```

Example 6: User defined function that does takes in multiple arguments

- The syntax of the function is:

```
void calculate_area(double A, double r)
{
     printf("Area of a circle with radius, %.3f m is %.3f
m^2\n",A,r);
    return;

}
```

Example 6: User defined function that does takes in multiple arguments

- Function prototype:

```
void calculate_area(double A, double r);
```

Example 6: User defined function that does takes in multiple arguments

```c
/*Lecture 8: Functions*/
/*Example 6: Calculate Area of a circle*/
#include <stdio.h>
#include <math.h>
#define PI 3.14159265359

void calculate_area(double A, double r);

int main(){
  double A, r;
  printf("Enter value of radius, r (m): "); /*prompt user*/
  scanf("%lf",&r); /*Assign user input to cariable r*/

  /*calculate Area*/
  calculate_area(A,r);

  return 0;
}

void calculate_area(double A, double r)
{
    A = PI*pow(r,2); /*A=r^2*/
    printf("Area of a circle with radius, %.3f m is %.3f m^2\n",r,A);
    return;
}
```

Function prototype

Main function

Calling function

User defined function

Conclusion

- Conclusion #1
- For each argument that is passed to the function, the parameter list must contain one entry, which specifies the type and the name.
    - Example 1: `void func1(int a, char b, double c)`
        - This function does not have a return value (`void`)
        - This function's name is `func1`
        - It takes in 3 arguments: type `int` named `a`, type `char` named `b` and type `double` named `c`.
    - Example 2: `double func2(float num)`
        - This function returns a variable of type `double`
        - This function's name is `func2`
        - It takes in 3 arguments: type `float` named `num`

Conclusion

- Conclusion #3
  - Some functions take no arguments, so the parameter list should be `void` or empty such as,

```
void funct5(void)
double funct6(void)
int  func7()
```

- Conclusion #4
  - Function prototype normally placed before `main()`
- Conclusion #5
  - Your function should be placed after `main()`.

Technical Informatics I

Lecture 8

Dr Fatimah