Lecture 4: Functions

Charudatt Kadolkar

Functions

- In mathematics, **functions** associate a value (an object from range set) to an every element of domain. We often say, *real* function or *complex* function, indicating the type of the value of the function.

  `sin(x), cos(x), exp(x)`

  etc.

- Ordinarily, we use the word function to indicate purpose of machine/working device. The function of the washing machine is to wash clothes. The function of the air conditioner is to cool the air. In these cases, there is no value associated with function. It indicates a piece to work.

- In C programming, we have already used mathematical functions like `sin(x)` . We have also used other kinds of functions, for example `printf()` prints formatted strings on the screen. `scanf()` reads the keyboard.

Area of Circle

```
float   r;

while   ( 1 > 0 ) {
     printf("Radius     (-1  to  stop   )?  ");
     scanf("%f",     &r);
     if  ( r < 0 ) break;
     printf("Area     = %f\n",   area(r));
}
```

However, in C, there is no such function called `area` . But C allows you to define your own.

Area of Circle

```
float   area(float    x)
{
     float   a;
     a = 4.0*atan(1.0)*x*x;
     return   a;
}

main()
{
     float   r;

     while   ( 1 > 0 ) {
          printf("Radius     (-1  to  stop   )?  ");
          scanf("%f",     &r);
          if  ( r < 0 ) break;
          printf("Area     = %f\n",    area(r));
     }
}
```

Area of Circle

```
float   area(float    x)
{
     float   a;
     a = 4.0*atan(1.0)*x*x;
     return   a;
}

main()
{
     float   r;

     while   ( 1 > 0 ) {
          printf("Radius     (-1  to  stop   )?  ");
          scanf("%f",     &r);
          if  ( r < 0 ) break;
          printf("Area     = %f\n",    area(r));
     }
}
```

Definition of the function

Area of Circle

```
float   area (float   x)
{
     float   a;
     a = 4.0*atan(1.0)*x*x;
     return   a;
}

main()
{
     float   r;

     while   ( 1 > 0 ) {
          printf("Radius     (-1  to  stop   )?  ");
          scanf("%f",     &r);
          if  ( r < 0 ) break;
          printf("Area     = %f\n",   area(r));
     }
}
```

Name of the function

Area of Circle

```
float   area( float   x )
{
     float   a;
     a = 4.0*atan(1.0)*x*x;
     return   a;
}

main()
{
     float   r;

     while   ( 1 > 0 ) {
          printf("Radius     (-1  to  stop   )?  ");
          scanf("%f",     &r);
          if  ( r < 0 ) break;
          printf("Area     = %f\n",    area(r));
     }
}
```

Parameter and its type (domain), enclosed in ()

Area of Circle

```
float area(float    x)
{
     float   a;
     a = 4.0*atan(1.0)*x*x;
     return   a;
}

main()
{
     float   r;

     while  ( 1 > 0 ) {
          printf("Radius     (-1  to  stop  )?  ");
          scanf("%f",    &r);
          if  ( r < 0 ) break;
          printf("Area    = %f\n",   area(r));
     }
}
```

type of the value of the function(return value)

Area of Circle

```
float   area(float    x)
 {
     float   a;
     a = 4.0*atan(1.0)*x*x;
     return   a;
 }

main()
{
     float   r;

     while   ( 1 > 0 ) {
          printf("Radius     (-1  to  stop   )?  ");
          scanf("%f",     &r);
          if  ( r < 0 ) break;
          printf("Area     = %f\n",    area(r));
     }
}
```

Body of the function is enclosed in brackets and contains declarations and statements

Function Definition

Syntax of a Function Definition is

```
Return-type    function-name(parameter declarations,    if any)
{
    declarations
    statements
}
```

The C program is only a collection function definitions and a few declarations. One of the functions must be called

`main` . When program is run, it begins by calling main function.

Prime numbers

Here, we want to define a function that detects prime numbers.

1. Name for a function?

   `is_prime`

2. What is the domain?

   positive integers.

3. What is the type of variable used as argument?

   `int`

4. What is the range?

   true, false

5. What is the return value and its type?

   0, 1, 2, `int`

   0 for true, 1 for false and 2 if argument is non-positive number.

Finally Definition will be

```
int  is_prime(int    m)
{ /* Body  of  the  function   */ ...  }
```

Prime numbers

```
#include   <stdio.h>
#include   <math.h>

int  isPrime(int    n);         /*  Prototype    */

/*  Main  functions    reads  a  number,   calls  is_prime   to  determine
    whether   the  number   is  prime   or  not,  prints   the  result
main()
{
}

/*  Determines    whether   num  is  prime.   return   0  if  true,   1  if  false,
    2  if  num  is  not  positive    */
int  isPrime(int    num)
{
}
```

Prime numbers

```
#include   <stdio.h>
#include   <math.h>

int  isPrime(int    n);         /*  Prototype    */

/*  Main  functions    reads   a  number,    calls   is_prime    to  determine
    whether    the   number    is  prime    or  not,   prints    the   result
main()
{
          int  result;
          int  m;
          printf("Enter     a positive    number    -> ");
          scanf("%d",&m);
          result   = isPrime(m);
          if  ( result    ==  0  ) printf("%d     is  a  prime   number\n",m);
          else
                    if  ( result    ==  1  ) printf("%d     is  not   a  prime   number\n",m);
                    else  printf("%d     is  not   positive\n",m);
          return;
}

/*  Determines     whether    num  is  prime.    return    0 if  true,   1 if  false,
    2  if  num  is  not   positive    */
int  isPrime(int    num)
```

Prime numbers

```
#include   <stdio.h>
#include   <math.h>

int  isPrime(int    n);

/*  Main  functions    reads   a  number,    calls   is_prime    to  determine
          :
          :

/*  Determines    whether   num  is  prime.   return   0 if  true,   1 if  false,
    2 if  num  is  not  positive    */
int  isPrime(int    num)
{
          int  i;
          if  ( num  <= 0 ) return   2;
          if  ( num  == 1 ) return   0;
          if  ( num  == 2 ) return   0;
          for  ( i = 2;  i <= (int)  sqrt(num);    i++  )
                    if  ( num  % i == 0 ) return   1;
          return   0;
}
```

Local Variables

```
#include   <stdio.h>
#include   <math.h>

int  isPrime(int    n);          /*  Prototype    */

/*  Main  functions    reads   a  number,    calls   is_prime    to  determine
    whether    the   number    is  prime    or   not,   prints    the   result
main()
{
          int  result;
          int  m;
          :
}

/*  Determines    whether    num   is  prime.    return    0  if  true,    1  if  false,
    2  if  num   is   not   positive    */
int  isPrime(int    num)
{
          int  i;
          :
}
```

result and m are local variables of main

num and i are local variables of is_prime

Passing Arguments

```
main()
{
        a = 2;  b = 5;
        swap(a,b);
        printf("a   = %d,  b = %d\n",   a,  b);
}

void  swap(int   x,  int  y)
{
        int  z;
        z = x;  x = y;  y = z;
        return;
}
```

This is called passing arguments by value.

Functions

Why use functions?

- **Divide and Conquer** Large complex jobs can be broken down to simple small tasks. It is easy to analyse and program the smaller tasks.
- **Reusability** A task, like calculating sine values, can be programmed once and reused by others, without worrying about method of implementation.
- **Debugging** It is much easier to test a small program.