BIRZEIT UNIVERSITY

Top-Down Design with Functions

Computer Science Department

Objectives

- To learn about functions and how to use them to write programs with separate modules
- To understand the capabilities of some standard functions in C
- To understand how control flows between function main and other functions
- To learn how to pass information to functions using input arguments
- To learn how to return a value from a function

Top-Down-Design

- A problem solving method
- First, break a problem up into its major sub problems
- Solve the sub problems to derive the solution to the original problem

Functions

- Definition:

  A function is a group of statements that together perform a task. Every C program has at least one function, which is **main()**, and all the most trivial programs can define additional functions

Functions

- Two types:
  1. C library functions (**sqrt (x), abs (x),…)**
  2. User defined functions (Your own functions)

Some Mathematical Functions

Function	Standard Header File	Example	Argument(s)	Result
abs(x)	<stdio.h>	x=-5 abs(x)=5	int	int
ceil(x)	<math.h>	x=45.23 ceil(x)=46	double	double
cos(x)	<math.h>	x=0.0 cos(x)=1.0	double (radians)	double
exp(x)	<math.h>	x=1.0 exp(x)=2.71828	double	double

Some Mathematical Functions

Function	Standard Header File	Example	Argument(s)	Result
fabs(x)	<math.h>	x=-8.432 fab(x)=8.432	double	double
floor(x)	<math.h>	x=45.23 floor(x)=45	double	double
log(x)	<math.h>	x=2.71828 log(x)=1.0	double	double
log10(x)	<math.h>	x=100.0 log10(x)=2.0	double	double

Some Mathematical Functions

Function	Standard Header File	Example	Argument(s)	Result
pow(x,y)	<math.h>	x=0.16 y=0.5 pow(x,y)=0.4	double double	double
sin(x)	<math.h>	x=1.5708 sin(x)=1.0	double (radians)	double
sqrt(x)	<math.h>	x=2.25 sqrt(x)=1.5	double	double
tan(x)	<math.h>	x=0.0 tan(x)=0.0	double (radians)	double

Functions – Example 1

Write a complete C Program to compute the following mathematical expression:

$x=b^2+c^2-2bc$

```
double x, b, c;
x= pow(b,2)+pow(c,2)-2*b*c;
```

Functions - Example 2

Write a complete C Program to compute the following mathematical expression:

$a^2=b^2+c^2-2bc \cos\alpha$, where α in degree

```
double a, b, c, alpha;
a=sqrt(pow(b,2)+pow(c,2) - 2 * b* c* cos(alpha * PI / 180.0));
```

converting from degrees to radians **is to simply** ***multiply*** **the number of** ***degree*** **by Π /180°**

User-Defined Functions

- Why Functions:

1) Useful for C programmers to divide their programs into separate functions (instead of big “chunk‘”). This make it *easy to debug the code and handling error*.

2) Reusability:

- Once a function is defined, it can be used over and over and over again.
- You can invoke the same function many times in your program.
- Use same function in several different (and separate) programs.

Functions

Types of functions:

1. Function with no arguments and no return value.
2. Function with no arguments but return value
3. Function with arguments and no return value
4. Function with argument and a return value

Functions

- Steps to write a function:
  1. Function Prototype
  2. Function Definition
  3. Function Call

1. Function Prototype

Tells the **compiler** about a *function's name*, *return type*, and *parameters*.

return_type function_name (parameter list)

Examples:

```
int  sum (int ,int );// with parameters and return value
void printNum (int);// with parameters and no return value
float area (); // no parameters and  with return value
double circumference (double);// with parameters and return value
void  printChar (char); // with parameters and no return value
void  printSquare();//no arguments and no return value
```

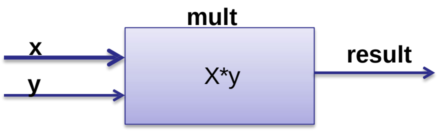
2. Function Definition

Provides the actual **body** of the function.

```
return_type  function_name ( parameter list )
{
        body of the function
}
```

Function Definition – Example 1

```
int mult( int x, int y)
{
      int result;
      result= x*y;
      return result;
}
```

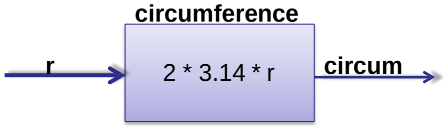


Function Definition – Example 2

```
void printNum ( int x)
{
        printf("%d", x);
}
```

Function Definition – Example 2

```
double circumference (double r)
  {
     double circum;
     circum= 2 * 3.14 * r;
     return circum;
  }
```



3. Function Call

- To **use** a function, you will have to call that function to perform the defined task.

Examples:

```
int mySum = mult(x,y);
double circum = circumference (r);
printNum(x);
```

Parameters

- a parameter is a special kind of variable, used in a function to refer to one of the pieces of data provided as input to the function.
- These pieces of data are called arguments
- Normal Variable vs. Parameter: these Arguments are defined *at the time of Calling Function*.
- Parameter Written In Function Definition is Called "Formal Parameter"
- Parameter Written In Function Call is Called "Actual Parameter".
- The parameter list refers to the type, order, and number of the parameters of a function.
- Parameters are optional; that is, a function may contain no parameters.

Formal & Actual Parameters

```
void main()
{
int num1;
display(num1);
}

void display(int para1)
{
-----------
-----------
}
```

- Here, this method is called "**call by value**".
- It copies the actual value of an argument into the formal parameter of the function.
- Changes made to the parameter inside the function have no effect on the argument.

- para1 is "Formal Parameter"
- num1 is "Actual Parameter"

Example: add two integers

```
#include <stdio.h>

int addNumbers(int a, int b);           // function prototype

int main()
{
      int n1,n2,sum;

      printf("Enters two numbers: ");
      scanf("%d %d",&n1,&n2);

      sum = addNumbers(n1, n2);          // function call

      printf("sum = %d",sum);

      return 0;
}

int addNumbers(int a,int b)           // function definition
{
      int result;
      result = a+b;
      return result;                      // return statement
}
```

Passing arguments to a function

```c
#include <stdio.h>

int addNumbers(int a, int b);

int main()
{
    ... .. ...

    sum = addNumbers(n1, n2);

    ... .. ...
}

int addNumbers(int a, int b)
{
    ... .. ...
    ... .. ...
}
```

Return Statement

- The return statement *terminates* the execution of a function and *returns* a value to the calling function.
- The program control is transferred to the calling function after return statement.

```c
#include <stdio.h>

int addNumbers(int a, int b);

int main()
{
    ... .. ...

    sum = addNumbers(n1, n2);

    ... .. ...
}

int addNumbers(int a, int b)
{
    ... .. ...
    return result;
}
```

sum = result

Example: Creating a void user defined function

```
#include <stdio.h>
void introduction()
|
int main()
{
     /*calling function*/
     introduction();
     return 0;
}
/* function return type is void and it doesn't have parameters*/
void introduction()
{
    printf("Hi\n");
    printf("My name is Chaitanya\n");
    printf("How are you?");
    /* There is no return statement inside this function, since its
     * return type is void
     */
}
```

Output:

```
Hi
My name is Chaitanya
How are you?
```

Functions (Exercises)

- Write a C program to compute the area of a circle with radius r. (Recall that $A=\pi r^2$.)
- In the same C program write a function to compute the circumference of a circle with radius r. (Recall that circum=2 * πr)

Area of a circle

```
#include <stdio.h>
#include <math.h>
#define PI 3.141
// 1. function prototype
double ComputeArea (double);

int main() {
   double r, area;

   printf("Enter the radius of the circle: \n");
    scanf("%lf",&r);
    area= computeArea(r); //  3. call function
    printf("The area of a circle with radius %5.3f is %S.3f.  \n",r,area);
    // Exit program.
    return 0;
    }

 // 2. Function Definition
double ComputeArea (double  r) {

     double arear;
     area = Pl*pow (r,2);
     return area;
     }
```

Area & Circumference of a circle

```
#include <stdio.h>
#include <math.h>
#define PI 3.141

double ComputeArea (double);
double circumference (double r);
int main() {
   double r, area;

   printf("Enter the radius of the circle: \n");
    scanf("%lf",&r);
    area= computeArea(r);
    double circum = circumference (r);
    printf("The area of a circle with radius %5.3f is %S.3f.  \n",r,area);
    printf("The circumference of a circle with radius %5.3f is %S.3f.
    \n",circum);
    return 0;
    }

double ComputeArea (double  r) {
     double arear;
     area = Pl*pow (r,2);
     return area;
     }
```

```
double circumference (double r)
{
  double circum;
   circum= 2 * 3.14 * r;
    return circum;
}
```

```c
#include <stdio.h>
void printNumber (int);
int main()
{
  int number;
  printf("please enter a number");
  scanf("%d",&number);
  printNumber (number);
  return 0;
}
void printNumber (int x)
{
  printf("%d",x);
}
```

```c
#include <stdio.h>
void printNumber ();
int main()
{

  printNumber ();
  return 0;
}
void printNumber()
{
  int number;
  printf("please enter a number");
  scanf("%d",&number);
  printf("%d",number);
}
```

Functions (more practice)

What will be the output if you execute the following C code?

```
#include <stdio.h>
int f(int , int , int );
int main ()
{
    int q;
    q = f(3, 3, 4);
    printf ("q is %d ", q);
}
int f(int q, int b, int c)
{
      int p;
      p = q * b + 2 * c;
      return (p);
}
```

Main function
q

f function
q=3 , b=3 , c=4
p=??

Output (screen):

q is 17

Functions (more practice)

What will be the output if you execute the following C code?

```
#include <stdio.h>
int f(int , int , int );
int main ()
{
    int q;
    q = f(3, 3, 4);
    printf ("q is %d ", q);
}
int f(int q, int b, int c)
{
      int p;
      p = q * b + 2 * c;
      return (p);
}
```

Main function
q

f function
q=3 , b=3 , c=4
p=??

Output (screen):

q is 17

Function Example

Write a C function that computes an *employee's gross salary*. **Given** are regular hours worked, overtime hours worked and hourly rate. Overtime hours are paid at 1.5 times an employee's normal hourly rate. These three values are stored in a separate file called "employee.txt". The main C program reads the values from this file, calls the **function** to compute the employee's gross salary and then prints the result on **screen**

Solution

```
#include <stdio.h>
double grossSalary(double, double, double)
int main(void)
{
  double reg_hours,      /* input   regular hours worked  */
         ot_hours,       /* input   overtime hours worked */
         rate,           /* input   hourly rate of pay    */
         gross;          /* output   gross salary         */

   FILE *ftp_in;
   ftp_in = fopen("employee.txt", "r" ) ;
   fscanf(ftp_in, "%lf %lf %lf", &reg_hours,&ot_hours ,&rate);
   gross = grossSalary(reg_hours,ot_hours ,rate);
   printf("\nThe gross salary is %.2f.\n", gross);
   fclose(ftp_in);
   return (0);
}
double grossSalary(double reg_hours, double  ot_hours, double reg_rate )
{
   return reg_hours * reg_rate + ot_hours * 1.5 * reg_rate;
}
```

```c
#include <stdio.h>          /* printf, scanf definitions */
#include <math.h>           /* pow definition */

/* Function prototype */
double scale(double x, int n);

int
main(void)
{
      double num_1;
      int num_2;

      /* Get values for num_1 and num_2 */
      printf("Enter a real number> ");
      scanf("%lf", &num_1);
      printf("Enter an integer> ");
      scanf("%d", &num_2);

      /* Call scale and display result. */
      printf("Result of call to function scale is %f\n",
             scale(num_1, num_2));

      return (0);
}

double
scale(double x, int n)
{
      double scale_factor;     /* local variable - 10 to power n */

      scale_factor = pow(10, n);

      return (x * scale_factor);
```

actual arguments

information flow

formal parameters

```
Enter a real number> 2.5
Enter an integer> -2
Result of call to function scale is 0.025
```

Functions (LAB)

Write a complete c program that asks the user to enter two numbers, finds and prints the sum of them. Your program should include at least one function called sum to return the sum of the two numbers.

Function prototype

int sum (int x, int y)

Question

A manufacturer wishes to determine the cost of producing an open-top cylindrical container. The surface area (المساحة الكلية) of the container is the sum of the area of the circular base plus the area of the outside ($\pi r^2+2\pi rh$).

Write a program to read the radius of the base r, the height of the container h, the cost per square centimeter of the material (cost) and the number of containers to be produced (Quantity) from a file called data.txt . *<u>You should calculate the cost of each container and the total cost of producing all the containers and print the results on screen.</u>*

Your program should include three functions:-

1)Calculate_Area which takes the radius and the height for the container and calculates the surface area .

2)Calculate_Cost which takes the area of the container and the cost per square centimeter of the material (cost) and calculates the cost of a single container

3)Calculate_Total which takes the cost of a single container and the number of containers (Quantity) and finds the cost of producing all the containers.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define PI 3.14
float Calculate_Area(float,float);
float Calculate_Cost(float,float);
float Calculate_Total(float,int);
int main()
{
    float r,h,cost,a,c1,c2;
    int n;
    FILE *in;
    in = fopen("data.txt","r");
    fscanf(in,"%f%f%f%d",&r,&h,&cost,&n);
    printf("The radius of the base is %.1f cm.\nThe hight of the container is %.1f cm.\nThe cost per squ
    a = Calculate_Area(r,h);
    c1 = Calculate_Cost(a,cost);
    c2 = Calculate_Total(c1,n);
    printf(" The Area = %.2f cm^2\n The cost for one container = %.2f $/container\n The cost per %d cont
    fclose(in);

    return 0;
}
```

```
float Calculate_Area(float r,float h)
{
    float result;
    result = PI * pow(r,2) + 2 * PI * r * h;
    return result;

}
float Calculate_Cost(float a,float cost)
{
    float result;
    result = a * cost;
    return result;

}
float Calculate_Total(float c1, int n)
{
    float result;
    result = c1 * n;
    return result ;
}
```