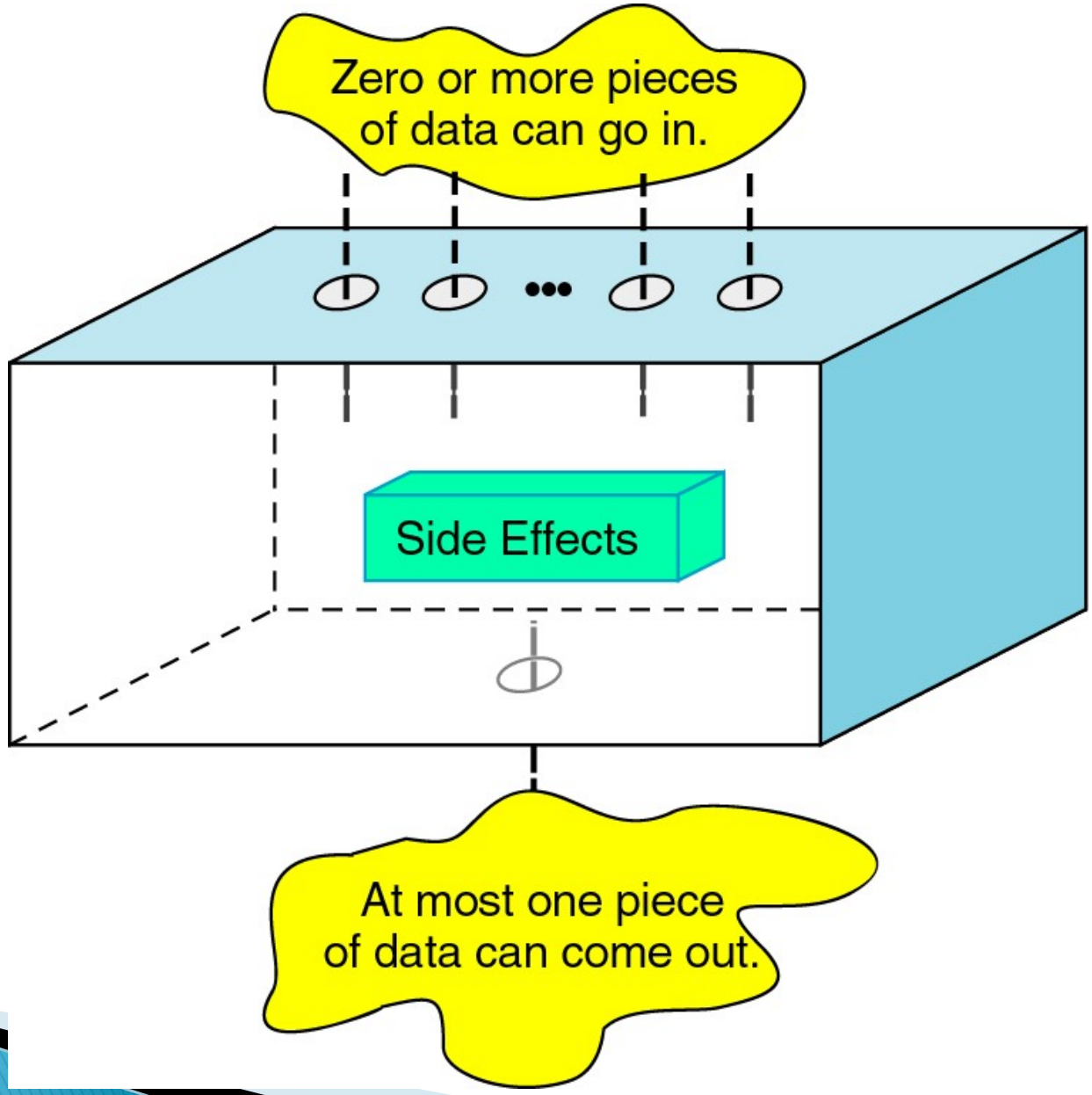
Functions


Zero or more pieces
of data can go in.
Side Effects
At most one piece
of data can come out.

Functions

- every C program must have a function called main
- program execution always begins with function main
- any other functions are subprograms and must be called

When a function is called,

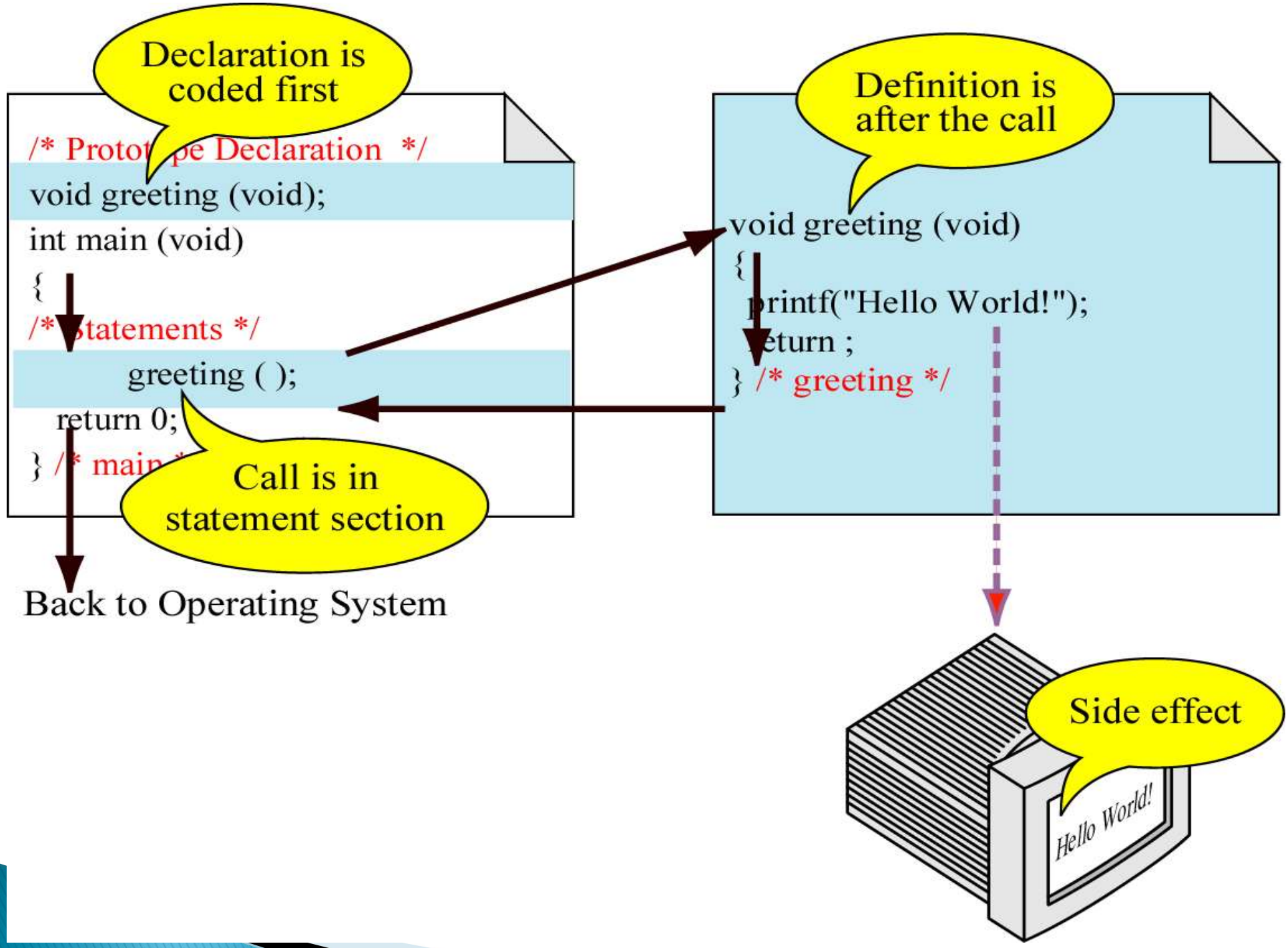
Then the flow of control passes to the first statement in the function's body. The called function's body statements are executed until one of these occurs:

return statement (with or without a return value),

or,

closing brace of function body.

Then control goes back to where the function was called.


Declaration is coded first
/* Prototype Declaration */
void greeting (void);
int main (void)
{
/* Statements */
greeting ( );
return 0;
} /* main */
Call is in statement section
Back to Operating System
Definition is after the call
void greeting (void)
{
printf("Hello World!");
return ;
} /* greeting */
Side effect
Hello World!

Function Concepts

- Function Prototype
- Function Call
- Function Definition

What is in a prototype?

A prototype looks like a heading but must end with a semicolon;
and its parameter list just needs to contain the type of each parameter.

```
int  Cube( int  );      /* prototype */
```

Function Call

- a function call temporarily transfers control to the called function's code
- when the function's code has finished executing, control is transferred back to the calling block

Function Call Syntax

FunctionName (Argument List)

The argument list is a way for functions to communicate with each other by passing information.

The argument list can contain 0, 1, or more arguments, separated by commas, depending on the function.

What is in a heading?

type of returned value

name of function

says no parameters

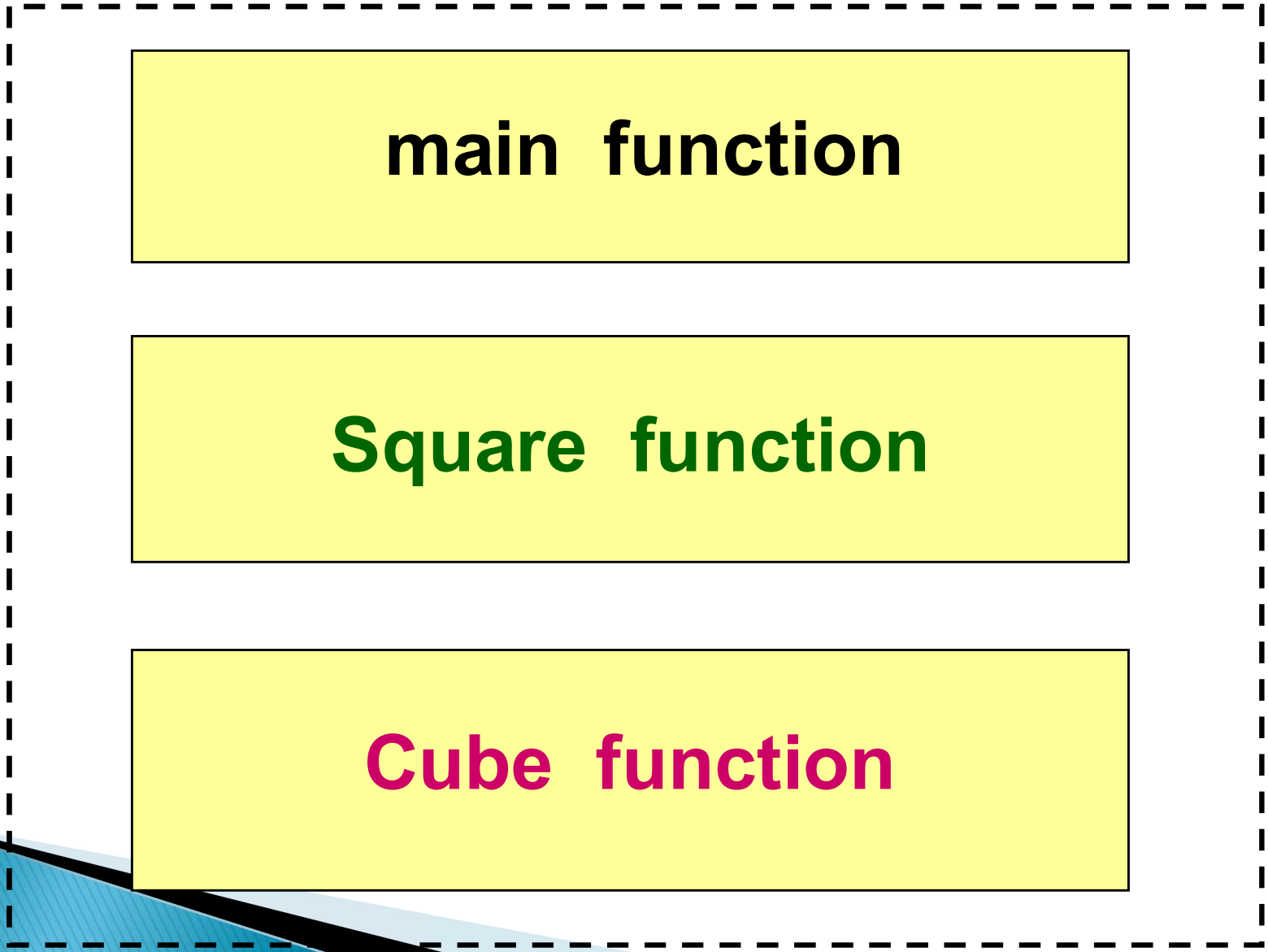
```
int main (        )

float calc ( float x )

int countchar ( String s )
```

HEADER FILE	FUNCTION	EXAMPLE OF CALL	VALUE
<stdlib.h>	abs(i)	abs(-6)	6
<math.h>	pow(x,y)	pow(2.0,3.0)	8.0
	fabs(x)	fabs(-6.4)	6.4
<math.h>	sqrt(x)	sqrt(100.0)	10.0
	sqrt(x)	sqrt(2.0)	1.41421
<math.h>	log(x)	log(2.0)	.693147
<stdio.h>	printf( )		

Program with Several Functions



Program with Three Functions

```
#include <stdio.h>

int Square( int );     /* declares these functions */
int Cube( int );

int main( )
{
      int num = 3 ;
      int sq , cb ;

      sq = Square(num);
      printf(" The square of 3 is %d \n ", sq );
      cb = Cube(num);
      printf(" The cube of 3 is %d \n ", cb );

      return 0;
}
```

Function Prototype: must be declared at top of program.

Function Call: invokes the function

Rest of Program

```
int Square( int n )
{
   int s = n * n ;
   return s;
}


int Cube( int n )
{
    int c = n * n * n ;
    return c;
}
```

Function Definition:
contains the actual function code.

A void function call stands alone

```
#include <stdio.h>

void  DisplayMessage ( int  n ) ;      /*declares function */

int main(  )
{
     DisplayMessage( 15 ) ;                /* function call */

     printf(  “Good Bye \n “ ) ;

     return 0 ;
}
```

A void function does NOT return a value

```
void  DisplayMessage ( int  n )
{
    printf( “ I have liked math for %d years \n ” , n );
}
```

Two Kinds of Functions

Value-Returning	Void
Always returns a single value to its caller and is called from within an expression.	Never returns a value to its caller, and is called as a separate statement.

Example

```
#include <stdio.h>
int Cube ( int ) ;                                  /* prototype */
void  main (  )
{
   int      yourNumber = 14;
   int      myNumber = 9 ;


printf(  “My Number = %d \n “ ,  myNumber ) ;
   printf( “ its cube is  %d \n “ , Cube (myNumber) ) ;

printf(  “Your Number = %d \n “ ,  yourNumber ) ;
   printf( “ its cube is  %d \n “ , Cube (yourNumber) ) ;

}
```

Write a void function

called DisplayMessage ( ) which you can call from main ( ) to describe the pollution index value it receives as a parameter.

Your city describes a pollution Index

less than 35 as "Pleasant",

35 through 60 as "Unpleasant",

and above 60 as "Health Hazard."

```
#include <stdio.h>
void DisplayMessage (int);          /* prototype */

int main ( )                        argument
{
        int pollutionIndex;

        printf(“ Enter pollution index : “);
        scanf( “ %d”, &pollutionIndex );
        DisplayMessage(pollutionIndex);   /* call */
        return 0;
}
```

parameter

```
void DisplayMessage( int index )
{
        if ( index < 35 )
                printf( “Pleasant” ) ;
        else if ( index <= 60 )
                printf(  “Unpleasant” ) ;
        else
                printf ( “Health Hazard” );
}
```

Parameter List

- The means used for a function to share information with the block containing the call

Classified by Location

Arguments	Parameters
Always appear in a function call within the calling block.	Always appear in the function heading, or function prototype.

Scope of Local and Global Variables