

ELEG 1043

Computer Applications in Engineering





Chapter 4: Selection Structures

C++ FOR ENGINEERS
AND SCIENTISTS ²

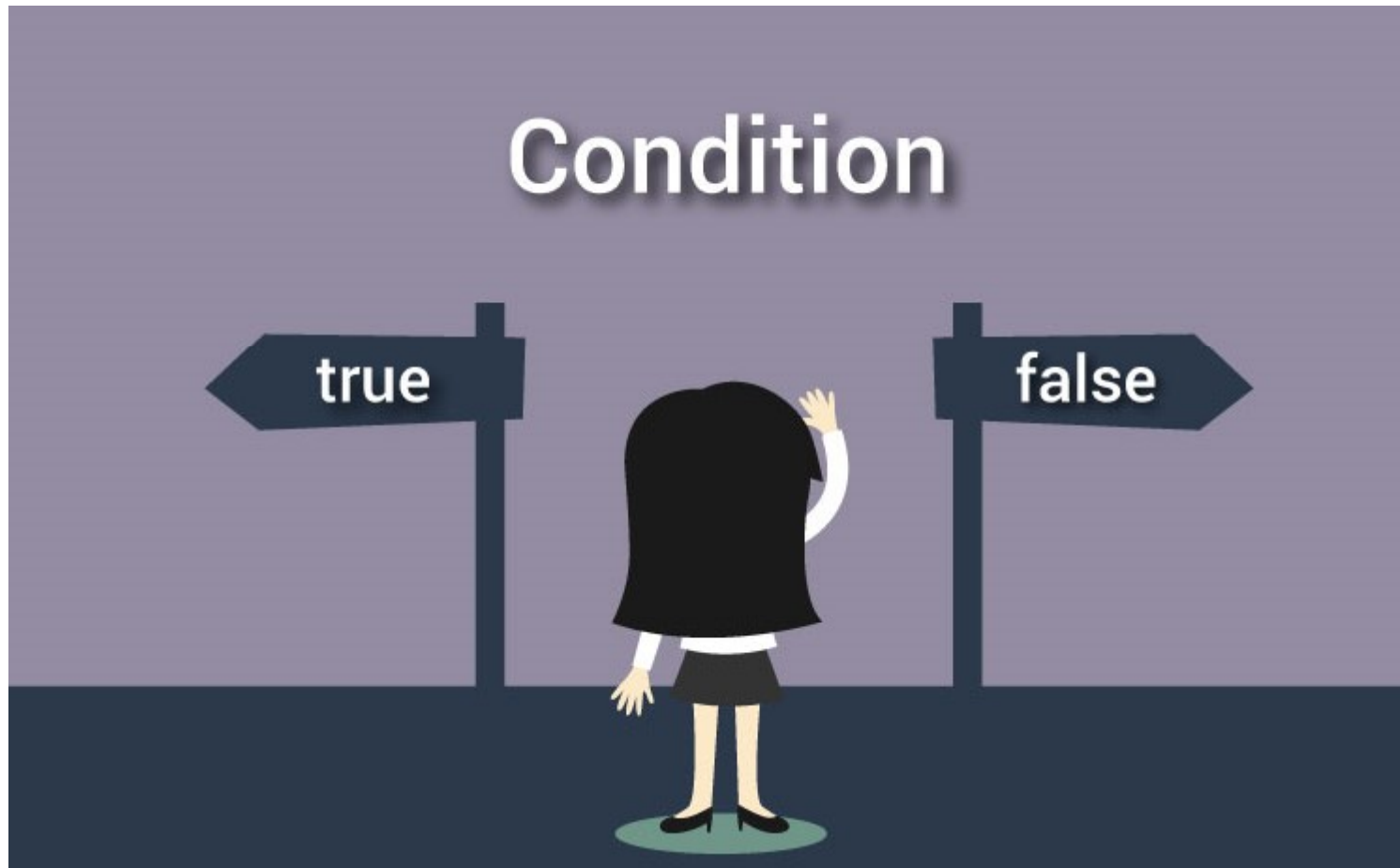
Acknowledgement

- Some of the slides or images are from various sources. The copyright of those materials belongs to their original owners.

Objectives

- In this chapter, you will learn about:
 - Selection criteria
 - The **if-else** statement
 - Nested **if** statements
 - The **switch** statement
 - Program testing
 - Common programming errors

Selection Criteria



<https://www.programiz.com/c-programming/c-if-else-statement>

Selection Criteria

- **if-else** statement: Implements a decision structure for two alternatives

Syntax:

if (condition)

statement executed if condition is true;

else

statement executed if condition is false;

Selection Criteria (continued)

- The condition is evaluated to its **numerical value**:
 - A **non-zero value** is considered to be **true**
 - A **zero** value is considered to be **false**
- The **else** portion is optional
 - **Executed only if the condition is false**

Logical Operators

- AND (&&): Condition is true only if **both expressions are true**
- OR (||): Condition is true if **either one or both of the expressions is true**
- NOT (!): Changes an expression to its opposite state; **true becomes false, false becomes true**

Logical Operators

- $A \ \&\& \ B \rightarrow$ True when both A and B are True, Otherwise False
- $A \ || \ B \rightarrow$ False when both A and B are False, Otherwise True

Exercise 1

1. (Practice) Determine the value of the following expressions, assuming $a = 5$, $b = 2$, $c = 4$, $d = 6$, and $e = 3$:

a. $a > b$

b. $a != b$

c. $d \% b == c \% b$

d. $a * c != d * b$

e. $d * b == c * e$

f. $!(a * b)$

g. $!(a \% b * c)$

h. $!(c \% b * a)$

i. $b \% c * a$

Exercise 1

2. (Practice) Using parentheses, rewrite the following expressions to indicate their order of evaluation correctly. Then evaluate each expression, assuming $a = 5$, $b = 2$, and $c = 4$.

a. $a \% b * c \&\& c \% b * a$

b. $a \% b * c || c \% b * a$

c. $b \% c * a \&\& a \% c * b$

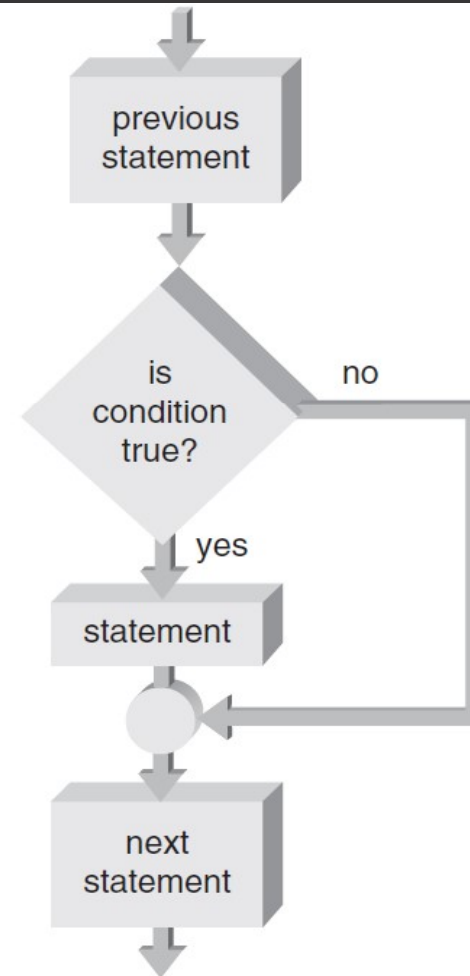
d. $b \% c * a || a \% c * b$

One-Way Selection

- **One-way selection:** An **if** statement without the optional **else** portion

```
int a = 1;  
if(a > 0)  
{  
    cout<<a;  
}
```

Figure 4.3 A one-way selection **if** statement



Nested `if` Statements

- `if-else` statement can contain any valid C++ statement, including another `if-else`
- Nested `if` statement: an `if-else` statement completely contained within another `if-else`
- Use braces to block code, especially when inner `if` statement does not have its own `else`

Exercise 2

Write a program that is to judge whether the number is an even number or an odd number, where **the number is input from keyboard.**

Example

```
#include <iostream>
using namespace std;

int main()
{
    int number;
    cout<<"Enter an integer: \n";
    cin>>number;

    // True if remainder is 0
    if( number%2 == 0 )
        cout<<number<<" is an even integer.\n";
    else
        cout<<number<<" is an odd integer.\n";

    return 0;
}
```

Exercise 3

- Write a program that is to judge if the number falls in the range **(-10, 0)**, where the number is received from keyboard.

Answer

```
#include <iostream>
using namespace std;

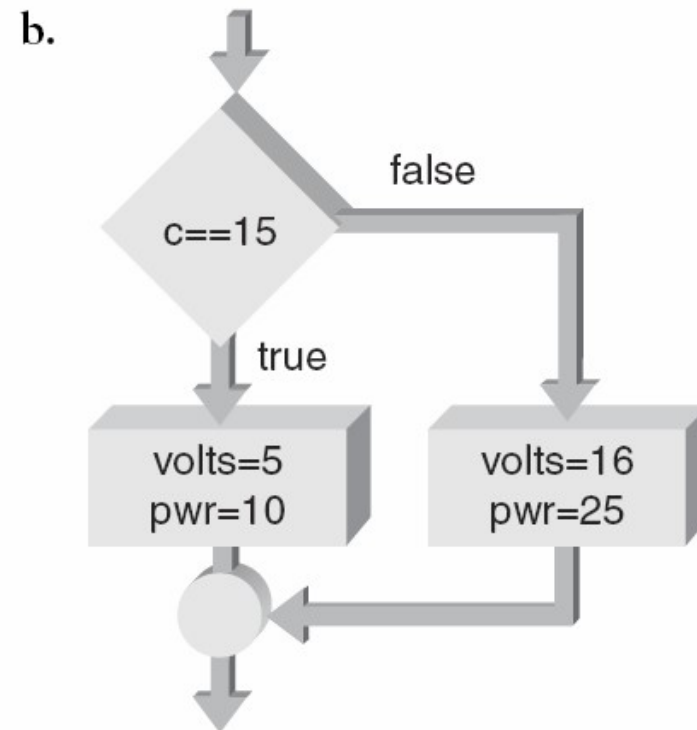
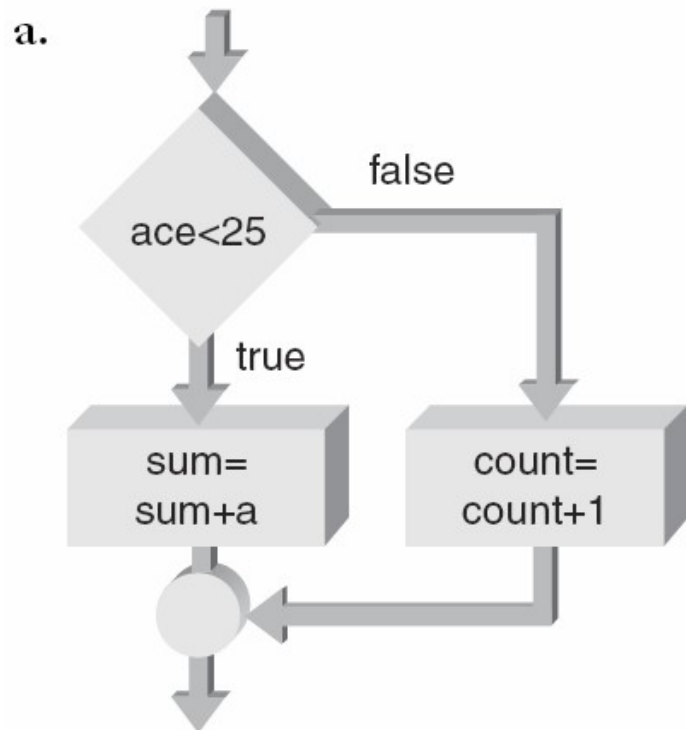
int main()
{
    int number;
    cout<<"Enter an integer: \n";
    cin>>number;

    // Test expression is true if number is less than 0
    if (number < 0)
    {
        if (number > -10)
            cout<<number<<" in the range (-10, 0)\n";
    }

    return 0;
}
```

Exercise 4

(Practice) Write `if` statements corresponding to the conditions illustrated in the following flowcharts:



Answer

```
a.  
if(a < 25)  
{  
    sum = sum + a;  
}  
else  
{  
    count = count + 1;  
}  
b.  
if(c == 15)  
{  
    volt = 5; power = 10;  
}  
else  
{  
    volt = 16; power = 25;  
}
```



Chapter 5: Repetition Statements

C++ FOR ENGINEERS
AND SCIENTISTS

Objectives

In this chapter, you will learn about:

- Basic loop structures
- **while** loops
- Interactive **while** loops
- **for** loops
- Loop programming techniques

Objectives (continued)

- Nested loops
- **do while** loops
- Common programming errors

Basic Loop Structures

- Repetition structure has **four** required elements:
 - Repetition statement
 - Condition to **be changed**
 - **Initial** value for the condition
 - Loop **termination**
- Repetition statements include:
 - **while**
 - **for**
 - **do while**

Example

- (a) Repetition statement
- (b) Condition to be changed
- (c) Initial value for the condition
- (d) Loop termination

```
        (c)      (d)      (b)
for(int i = 0; i < 10; i++)
{ cout<<i<<endl; (a)
}
```


Basic Loop Structures (continued)

- The **condition** can be tested
 - At the beginning: **Pretest** or **entrance-controlled** loop
 - At the end: **Posttest** or **exit-controlled** loop
- Something in the loop body must cause the condition to **change**, to avoid an **infinite loop**, which **never terminates**

while Loops



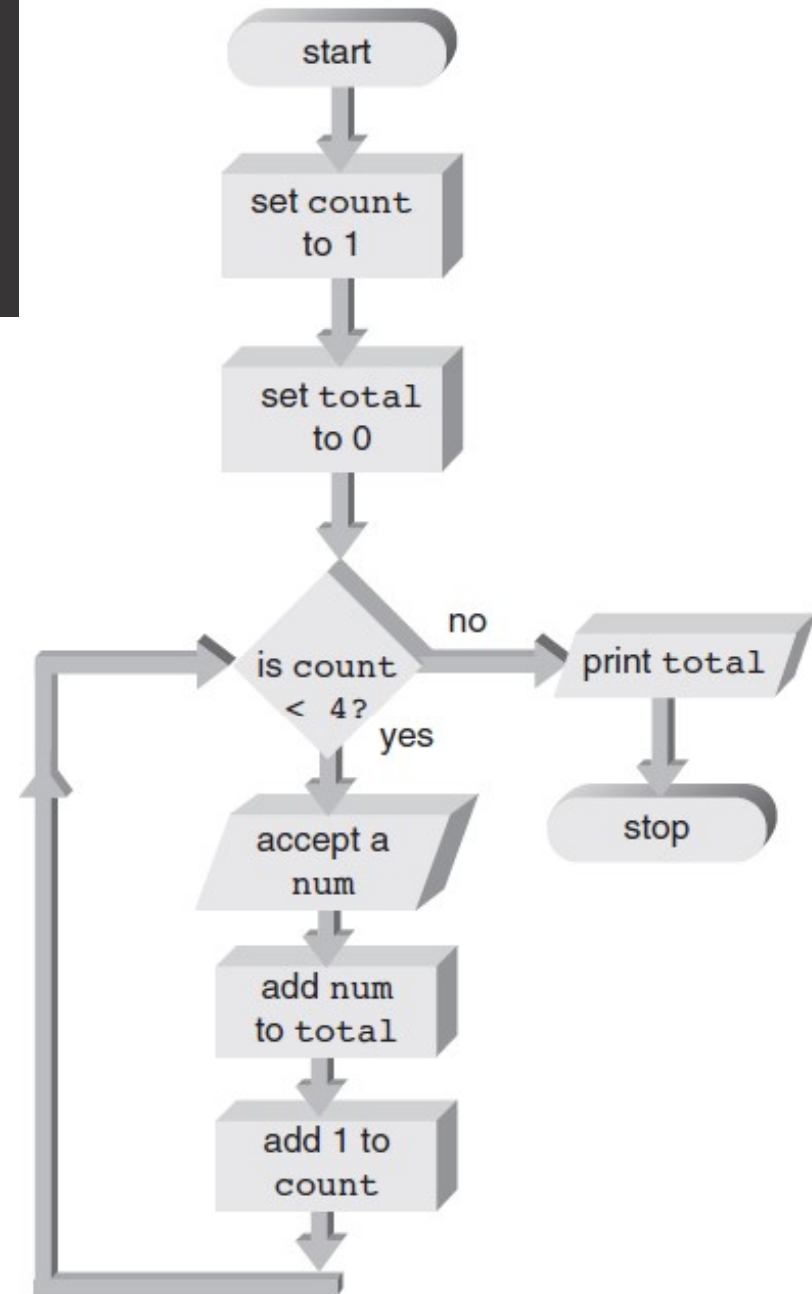
while Loops

- **while statement** is used to create a `while` loop
 - Syntax:
while (expression)
statement;
- Statements following the expressions are executed as long as the expression condition remains true (evaluates to a non-zero value)

Interactive `while` Loops

- Combining interactive data entry with the `while` statement provides for repetitive entry and accumulation of totals

Interactive while Loops (cont'd)



Interactive `while` Loops (cont'd)

```
#include <iostream>
using namespace std;

int main()
{
    int count = 0, total = 0;

    while(count < 4)
    {
        int num;
        cout<<"Enter a num \n";
        cin>>num;
        total = total + num;
        count = count + 1;
    }

    cout<<"The total is "<<total;
    return 0;
}
```

break and continue Statements

- **break** statement
 - Forces an immediate break, or exit, from **switch**, **while**, **for**, and **do-while** statements
 - Useful for breaking out of loops when an unusual condition is detected

break and continue Statements (cont'd)

- Example of a break statement:

```
while (count <= 10)
{
    cout << "Enter a number: ";
    cin >> num;
    if (num > 76)
    {
        cout << "You lose!\n";
        break;           // break out of the loop
    }
    else
        cout << "Keep on trucking!\n";
    count++;
}
// break jumps to here
```


Exercise 4

- Write a Program to print the numbers from 1 to 10 in increments of 3. The output of your program should be the following:

1 4 7 10

Answer

```
#include <stream>
using namespace std;
int main(){
int num = 1;
while(num < 11){
    cout<<num<<" ";
    num = num + 3;}
return 0;
}
```

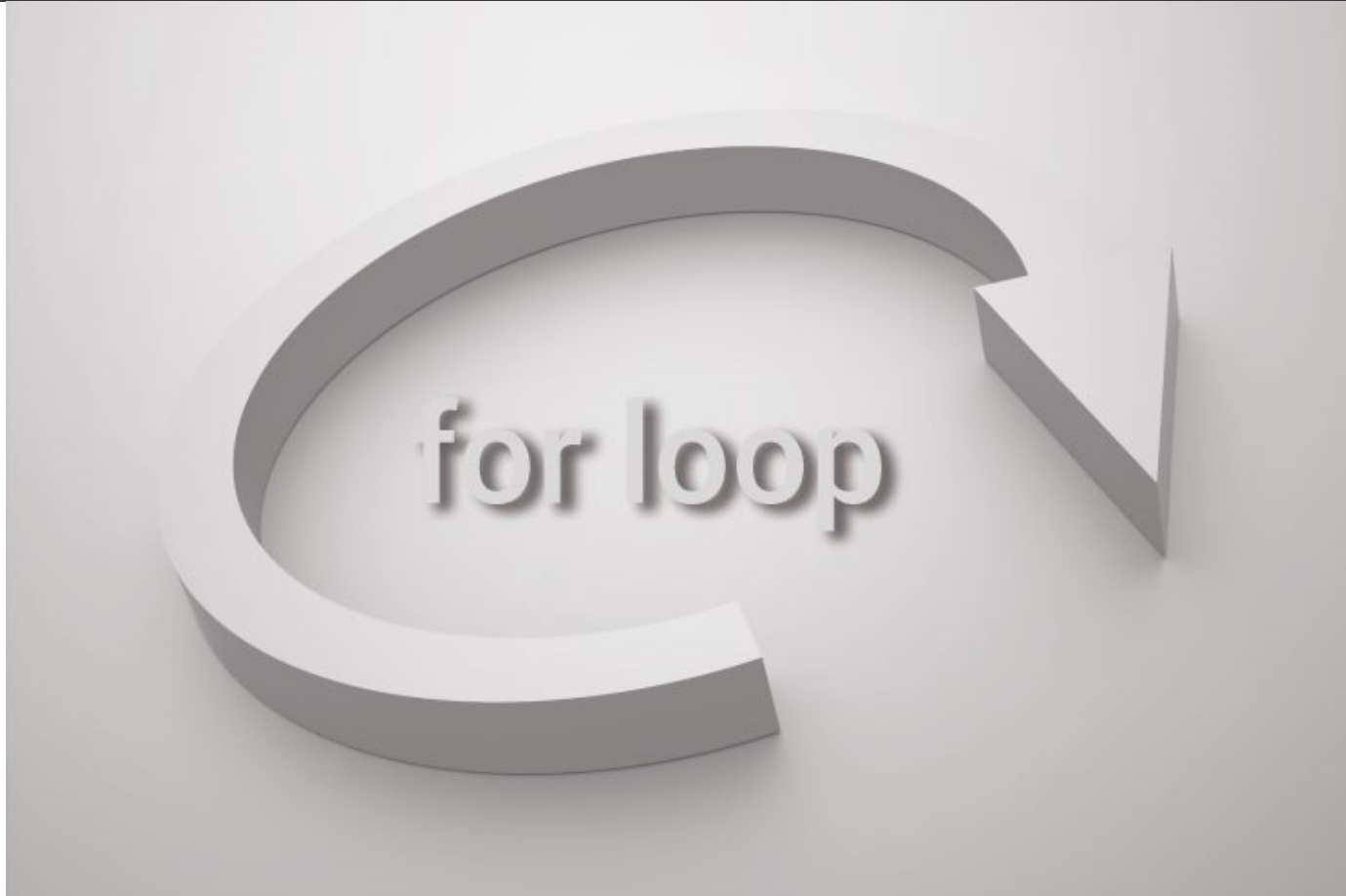
Exercise 5

- Write a Program that is to produce a table starting at a Celsius value of 2 and ending with a Celsius value of 60, in increments of 10 degrees.

Answer

```
#include <stream>
using namespace std;
int main(){
int celsiusValue= 2;
while(celsiusValue < 61){
    cout<< celsiusValue <<" ";
    celsiusValue = celsiusValue + 10;}
return 0;
}
```

for Loops



for Loops

- **for** statement: A loop with a fixed count condition that handles alteration of the condition
 - Syntax:
for (initializing list; expression; altering list)
statement;
- **Initializing list:** Sets the starting value of a counter
- **Expression:** Contains the maximum or minimum value the counter can have; determines when the loop is finished

Exercise 6

(Desk check) Determine the value in `total` after each of the following loops is executed:

a. `total = 0;`

```
for (i = 1; i <= 10; i = i + 1)
```

```
total = total + 1;
```

b. `total = 1;`

```
for (count = 1; count <= 10; count = count + 1)
```

```
total = total * 2;
```

c. `total = 0;`

```
for (i = 10; i <= 15; i = i + 1)
```

```
total = total + i;
```

d. `total = 50;`

```
for (i = 1; i <= 10; i = i + 1)
```

```
total = total - i;
```

Answer

a. $\text{total} = 0 + 1 \cdot 10 = 10$

b. $\text{total} = 1 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 1,024$

c. $\text{total} = 0 + 10 + 11 + 12 + 13 + 14 + 15 = 75$

d. $\text{total} = 50 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 = -5$



Chapter 6: Modularity Using Functions

C++ FOR ENGINEERS
AND SCIENTISTS

Defining a Function

- Every C++ function consists of two parts:
 - **Function header**
 - **Function body**
- Function header's purpose:
 - **Identify data type** of value function returns, **provide** function with **name**, and **specify** number, order, and type of **arguments** function expects
- Function body's purpose:
 - To operate on passed data and return, **at most**, one value directly back to the calling function

Exercise 7

1. (Practice) For the following function headers, determine the number, type, and order (sequence) of the values that must be passed to the function:
 - a. `void factorial(int n)`
 - b. `void volts(int res, double induct, double cap)`
 - c. `void power(int type, double induct, double cap)`
 - d. `void flag(char type, double current, double time)`
 - e. `void total(double amount, double rate)`

Answer

- a. Require **one** int value
- b. Require **three** values in this order: an int, a double, and a double
- c. Require **three** values in this order: an int, a double, and a double
- d. Require **three** values in this order: a char, a double, and a double
- e. Require **two** values in this order: a double, and a double

Exercise 8

4. (Statics) A beam's second moment of inertia, also known as its area moment of inertia, is used to determine its resistance to bending and deflection. For a rectangular beam (see Figure 6.6), the second moment of inertia is given by this formula:

$$I = b \times h^3 / 12$$

I is the second moment of inertia (m^4).

b is the base (m).

h is the height (m).

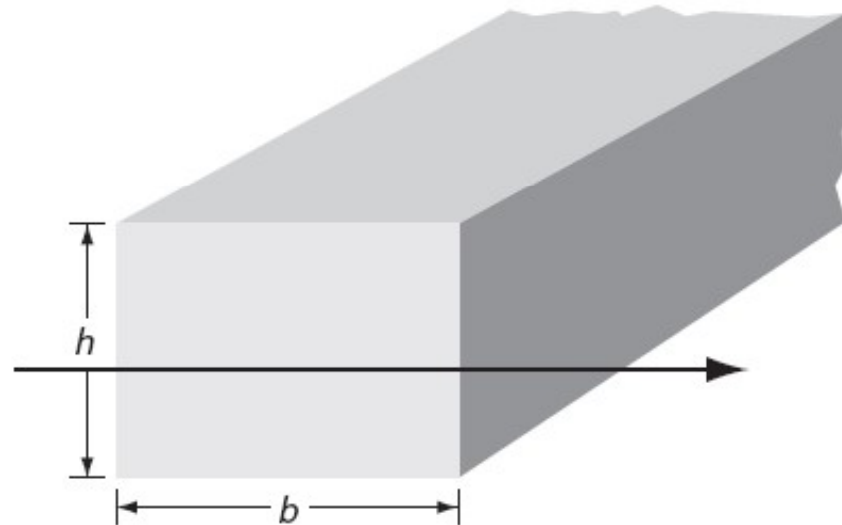


Figure 6.6 Calculating a beam's second moment of inertia

- a. Using this formula, write a function called `beamMoment ()` that accepts two double-precision numbers as parameters (one for the base and one for the height), calculates the corresponding second moment of inertia, and displays the result.

Answer

```
double beamMoment(double b, double h)
{
    double I = (b*h*h*h)/12.0;
    cout<<"The second moment is "<<I<<endl;
    return I;
}
int main(){
    double b = 0.2, h = 1.2;
    double I = beamMonment(b,h);
    return 0;}
```

Exercise 9

3. (Practice) Write function headers for the following:

- a. A function named `check()` that has three parameters. The first parameter should accept an integer number, and the second and third parameters should accept a double-precision number. The function returns no value.
- b. A function named `findAbs()` that accepts a double-precision number passed to it and returns its absolute value.
- c. A function named `mult()` that accepts two floating-point numbers as parameters, multiplies these two numbers, and returns the result.
- d. A function named `sqrIt()` that computes and returns the square of the integer value passed to it.
- e. A function named `powfun()` that raises an integer number passed to it (as an argument) to a positive integer power and returns the result as an integer.
- f. A function that produces a table of the numbers from 1 to 10, their squares, and their cubes. No arguments are to be passed to the function, and the function returns no value.

Answer

- a. `void check(int a, double b, double c);`
- b. `double findAbs(double num);`
- c. `float mult(float num1, float num2);`
- d. `int sqrtIt(int num);`
- e. `int powfun(int num, int power);`
- f. `void displayTable(void);`