Functions

1E3
Topic 9

Objectives

- This topic should allow students to
  - Understand the importance of abstraction in programming
  - Recognise when a function would be useful.
  - Design appropriate functions.
  - Understand how functions are used in C++.
  - Write and use functions in C++.
- Read chapter 6 of the textbook now.
Sample Function “cube”

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

double cube (double x);  // Function declaration

int main () {
    double x, x3;
    x = 4;
    x3 = cube (x);  // Use of the function
    cout << x << " cubed is " << x3;
    return 0;
}

double cube (double x) { 
    return (x*x*x);  // Function definition
}
```

See also

- Previous slide came from cube.cpp which shows another example of using the cube fn:
  - `cout << "\n5 cubed is " << cube(5);`
- fftocmfn.cpp is your first practical but using a function to do the calculation:
  - `double ft_to_cm (int ft, int in) {
      return ((ft*12)+in) * 2.54; }
`
Function Declaration

- Declare functions before use
- Declaration includes all you need to know to use the function
  - double cube (double x);
  - double VAT_added_price (double price, double VAT_rate);
  - double ppsi (double diam, double price);
- Provide a comment with each function declaration.

Function Definition

- Function definition consists of a declaration and the function body
- The body
  - computes the function value
  - returns the value
    - return (x*x*x);
    - return (price / area);
- (Note that we are ignoring void functions for now.)
Using the function

- A function call can appear anywhere an expression of the type returned by the function is expected:
  - double x = ppsi (10, 15);
  - cout << ppsi (8, 7.99);
- All our examples so far return doubles, but functions can return ints, chars, strings, booleans.

Functions of other types

- int max (int x, int y, int z);
  - Use:
    - n = max (10, p, q);
- string letter_grade (int mark);
  - Returns D, P1, P2, ..., F1, F2 ..., based on the percentage grade.
  - Use:
    - cout << letter_grade (55) << endl;
**Boolean Functions**

- Boolean functions are useful for abstracting complex conditions.

- `bool even (int n);`
  - returns true if n is an even number
  - Use:
    ```
    bool ok = even (n/2);
    if (even (number)) . . .
    ```

  - Defn body: `{ return (n%2 == 0); }

---

**Another Boolean Function**

- Declaration:
  - `bool is_a_factor_of (int n, int f);`

- Use:
  - if (is_a_factor_of (number, i))
    - composite = true;

- Definition:
  - `bool is_a_factor_of (int n, int f)`
    - `{ return ((n % f) == 0); }

- Useful in testing for primes or perfect numbers
**Parameters and arguments**

```c
double ppsi (double d, double p)
{
    double area = d/2 * d/2 * PI;
    return (p / area);
}
```

- `d` and `p` are **formal parameters** of the function.
- In the call
  
  ```c
  x = ppsi (diameter1, price1)
  ```

  - the **values** of `diameter1` and `price1` are the arguments passed to the function.

**Parameter-passing**

- Suppose `diameter1` is 10 and `price1` is 15.
- 10 is passed as the first argument, and is the value assigned to the formal parameter `d`.
- 15 is passed as the second argument, and is the value assigned to `p`.
- It’s as though the call is
  ```c
  ppsi (10, 15)
  ```

  - Inside the body of the function `d` and `p` are like variables whose initial values are 10 and 15 respectively. (see diagrams in lecture).
Zero Parameters (advanced)

- So far we’ve seen functions that take 1 or more parameters:
  - bool even (int n);
  - double ppsi (double diameter, double price);
  - int max (int x, int y, int z);
- What would it mean to have a function with no parameters?
  - int f ();
  - double g ();
- (Once we introduce void functions there will be more reasons to have zero parameters!)

In Class Programming

- Let’s write a payroll program (details on web)
  - Use a top-down design approach.
  - i.e. write the overall algorithm first
  - Worry about the details of individual tasks later.
  - These individual tasks can be handled by functions.
Algorithm for payroll

- For each record in the file
  - Read the details
  - Compute the gross pay
    - Based on hours and hourly rate
  - Compute the tax due
    - Based on gross pay and tax_credit
  - Compute the net pay
  - Output the computed details

When to use a function

- When the task is easy to identify but hard to implement or hard to understand the code.
- Whenever you are about to write the same or very similar code a second time
  - E.g. ppsi formula in our better-value pizza program
  - See pizzafn.cpp
```cpp
#include <iostream>
using namespace std;

double f (double x, int y);

int main (){
    cout << f(10,4) << "\n";
    return 0;
}

double f (double x, int y) {
    if (y==0)
        return 1;
    else
        return (x * f(x, y-1));
}
```

**ADVANCED EXERCISE**

- What does this program print?
- What does the function f compute?