
Functions

Divide and Conquer

Last Time

- We
 - Completed discussing reading from and writing to files
- Today we will
 - Begin learning about functions and modularity in C++

Functions

- Functions are a way of building modules in your program
- Encapsulate some calculation
- Less repetitive code
- Example:
 - `x = sqrt(y);`
 - `cout << x << endl;`

Library functions

- Code reuse. Why reinvent the wheel?
- Libraries are collections of often used functions
- Math library is an example
 - `#include <cmath>`
- List of functions given on p. 173
- We can call (invoke) the functions as follows:
 - `abs(x)`
 - `pow(x, y)`
 - `sin(x)`

17.1 Problem

- How would we rewrite the powers of 2 problem using the `cmath` library? (Output first 10 powers of 2)

Programmer Defined Functions

- We can write our own functions
- Implement top down design
 - Determine major steps of program
 - Write functions to implement each step
 - Program is more modular--each step is clearly defined
 - Details of steps are hidden

Example

- Write a program that outputs the cube of the first ten integers

10/26/05

CS150 Introduction to Computer Science 1

7

Example

```
#include <iostream>
int cube(int);
void main()
{
    int i, cubeOfI;
    for (i = 1; i <= 10; i++)
    {
        cubeOfI = cube(i);
        cout << "Cube of " << i << " is " << cubeOfI << endl;
    }
}
int cube (int k)
{
    return k*k*k;
}
```

10/26/05

CS150 Introduction to Computer Science 1

8

Working with Functions

- **1. Function prototypes**
- Function must be declared before it is referenced
- General form:
 - `type fname(datatypes of formal_arguments);`
- Example:
 - `int cube(int);`

10/26/05

CS150 Introduction to Computer Science 1

9

Working with Functions

- **2. Function definitions**
- General form:

```
type fname(formal_arguments)
{
    local_declarations;
    statements;
}
```
- Example:

```
int cube(int k)
{
    return k*k*k;
}
```

10/26/05

CS150 Introduction to Computer Science 1

10

Working with Functions

- **3. Function call**
- General form:
 - `fname(actual_arguments);`
- Example:
 - `cubeOfI = cube(i);`

10/26/05

CS150 Introduction to Computer Science 1

11

Things to remember

- Formal arguments are the arguments in the function definition
- Actual arguments are the values or variables in the function call
- Order of the arguments is very important!
- Make sure types match

10/26/05

CS150 Introduction to Computer Science 1

12

17.2 Problem

- Write a function that sums integers in a particular range
- Write a program that uses the above function

10/26/05

CS150 Introduction to Computer Science 1

13

Example

```
#include <iostream>
char isEven(int);
int getVal();
int main(void)
{
    int num;
    while ((num = getVal()) != -999)
    {
        if (isEven(num) == 'E')
            cout << "EVEN" << endl;
        else
            cout << "ODD" << endl;
    }
    return 0;
}
```

10/26/05

CS150 Introduction to Computer Science 1

14

Example (cont.)

```
char isEven(int number)
{
    const char even = 'E';
    const char odd = 'O';
    char answer;
    if (number % 2 == 0)
        answer = even;
    else
        answer = odd;
    return answer;
}
```

10/26/05

CS150 Introduction to Computer Science 1

15

Example (cont.)

```
int getVal()
{
    int number;
    cout << "Enter an integer number: ";
    cin >> number;
    return number;
}
```

10/26/05

CS150 Introduction to Computer Science 1

16

Summary

- In today's lecture we covered
 - Library functions
 - Programmer defined functions
- Readings
 - P. 170 - 180

10/26/05

CS150 Introduction to Computer Science 1

17