Advanced Function Concepts

CS 2308
Fall 2012
Jill Seaman

Overloading Functions

- Overloaded functions have the same name but different parameter lists.
- Used to create functions that perform the same task over different sets of arguments.
- The parameter lists of each overloaded function must have different types and/or number of parameters.
- Compiler will determine which version of the function to call based on arguments and parameter lists.

Example: Overloaded functions

```cpp
double calcWeeklyPay (int hours, double payRate) {
    return hours * payRate;
}
double calcWeeklyPay (double annSalary) {
    return annSalary / 52;
}

int main () {
    int h;
    double r;
    cout << "Enter hours worked and pay rate: ";
    cin >> h >> r;
    cout << "Pay is: " << calcWeeklyPay(h,r) << endl;
    cout << "Enter annual salary: ";
    cin >> r;
    cout << "Pay is: " << calcWeeklyPay(r) << endl;
    return 0;
}
```

Output:
Enter hours worked and pay rate: 37 19.5
Pay is: 721.5
Enter annual salary: 75000
Pay is: 1442.31

See calcpay2.cpp for a more complete example

Example: Overloaded functions prototypes

- Different number of arguments:
  ```
  int sum (int, int);
  int sum (int, int, int);
  int sum (int, int, int, int);
  ```

- Different types:
  ```
  string getString (int);
  string getString (double);
  string getString (char);
  ```
Default Arguments

- A default argument is a value assigned to the parameter when its argument is not provided in the function call.
- The default argument is usually listed in the function prototype:

```c
int showArea (double = 20.0, double = 10.0);
```
- Default arguments are literals or constants with an = in front of them, occurring after the data types listed in a function prototype.

Example: Default Arguments

```c
void displayStars(int = 10, int = 1);  
int main () {  
    displayStars();  // uses 10 x 1  
    cout << endl;  
    displayStars(5);  // uses 5 x 1  
    cout << endl;  
    displayStars(7, 3); // uses 7 x 3  
    return 0;  
}  
void displayStars(int cols, int rows) {  
    for (int down = 0; down < rows; down++) {  
        for (int across = 0; across < cols; across++)  
            cout << "*";  
        cout << endl;  
    }  
}  
```

Output:
```
**********  
*****  
*******  
*******  
*******  
```

Default Arguments

- When an argument is left out of a function call, all arguments that come after it must be left out as well.

```c
displayStars(5);  // uses 5 x 1  
displayStars(7);  // NO, won't work for 10 x 7  
```

- If not all parameters to a function have default values in the prototype, the parameters with defaults must come last:

```c
int showArea (double = 20.0, double = 10.0);  //NO  
int showArea (double, double = 20.0);  //OK  
```
Stubs and Drivers

- Useful for testing and debugging program and function logic and design
- Stub: A dummy function used in place of an actual function as a temporary placeholder
- Usually displays a message indicating it was called. May also display parameters

```cpp
void processList(int values[], int size) {
    cout << "Inside ProcessList, unfinished.\n";
}
```

Stubs and Drivers

- Driver: A function or program that tests another function by calling it, usually with constant values
- Various arguments are passed and return values are tested
- Usually no input from user

```cpp
int main() {
    int testList1[] = {3,4,2,6,7,10};
    int testList2[] = {7};
    processList(testList1,6);
    showArray(testList1,6);
    processList(testList2,1);
    showArray(testList1,1);
}
```