Notes #6

Functions
Part 1
Chapter 9

CMPT 130
© Dr. B. Fraser
Topics

1) How can we break up a program into smaller sections?
2) How can we pass information to and from functions?
3) How long do variables exist for?
Functions
Functions

- Functions:
  break a program down into manageable pieces.
- Each function should perform... one specific task.
- Also called methods, or procedures.
- Ex:
  - calculate a value, display the menu.
- Allows for the divide and conquer approach:
  - Divide: split the big problem down into multiple smaller problems.
  - Conquer:...solve each smaller problem.
// A simple C++ program.
#include <iostream>
using namespace std;

void displayMsg() {
    cout << "Hello world\n";
}

int main() {
    displayMsg();
    return 0;
}
Function definition

- A function (like a variable) must be defined before it can be called.
  - For the moment, put the definition of a function earlier (above) in the file than any calls to the function; otherwise will not compile.

- Function Return Type:
  - a specific type (such as int or bool or char); or
  - no value at all, by using the type void.
Review

• What is the difference between defining a function and calling a function?

• Write a function to display "I code therefore I am."
Getting data in and out of a function.
# Function Parameters

## Function call (use):
```cpp
int main( ) {
    displayNTimes("hi", 5);
    return 0;
}
```

- **Arguments**: ..
  Values passed to a function.

- **Parameter List**: ..
  Type & name for each parameter.
  - Inside the (…) of the function header.
  - May be empty if no parameters required.

- **Parameters**: ..
  Variables which hold the values of arguments.
  - These are variables inside the method.

## Function definition:
```cpp
void displayNTimes(string msg, int n) {
    while (n > 0) {
        cout << msg << endl;
        n--;
    }
}
```
Returning a value

• The return statements does 2 things:
  – Causes the current function to exit, returning control to the calling function.
  – Passes back a value from the function.

/*
Return the number of points the user scored based on the number of zombies killed.
Returns 0 if number killed is less than 0.
*/

int calcScore(int numZombies)
{
    if (numZombies < 0) {
        return 0;
    }
    return numZombies * POINTS_PER_ZOMBIE;
}
Returning a value vs Printing a value

• When a function calculates a value, it usually returns it, not prints it.

• Analogy:
  − You are voting in a referendum on a mail-in ballot, mailed to you by Elections Canada.
  − Do you say your vote aloud, or return your ballot to Elections Canada?

```
//... Return vote
int getVote()
{
    return 1;
}

//... Say your vote
void getVote()
{
    cout << 1 << endl;
}
```
• Write a function:
  - named add()
  - which accepts 2 int parameters; and
  - returns the sum of the two parameters as an int
Local vs Global Scope
Local variables

• Local variables:
  Variable declared inside a function.
  − Restricted scope (visibility) to within the function.
  − Restricted lifetime to when function is executing.
  − (These Includes function parameters.)

• What's that mean?
  − Cannot use a local variable outside the function.
  − Local variables are...
    destroyed when function ends.
    Next time through, a new one is created.
Global variables

- Global variables are declared outside of all functions.
  - Accessible anywhere between its definition and the end of the .cpp file.
  - Lifetime is the same as the program.

- Guidelines:
  - Good for constants:
    ```cpp
    const int DAYS_PER_WEEK = 7;
    ```
  - Often problematic for variables: can be very hard to understand and debug global variables.

- Use local variables as much as possible.
Scope and variable names

- Scope is the part of the program where a variable can be accessed.
  - Global scope: outside of all other scopes.
  - Local scope to a function: Inside a function.
  - Blocks: Any block {...}, such as for a while loop.

- You could reuse a variable name in different nested scopes, but is very confusing!
  - Try and give variables in nested scope unique names.
## Scope, Lifetime, and Puppies

<table>
<thead>
<tr>
<th>Lifetime</th>
<th>Local</th>
<th>Global</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temporary</td>
<td>Local variable</td>
<td></td>
</tr>
<tr>
<td>Persistent</td>
<td>Static local variable</td>
<td>Global variable</td>
</tr>
</tbody>
</table>
Summary

• Function definition: type, name, parameter list, body.
• Function call must use (): int age = getAge();
• Use return to pass back a value.
• Scope
  - Local variables exist only inside the function.
  - Global variables often bad; global constants good
Practise Review Questions

• Write just function headings (no body) for the following:
  – apple(): takes two ints, returns a float.
  – orange(): takes two ints and prints out the sum.

• Write a function named max() which:
  – Accepts two int values
  – Returns the maximum of the two.

• Write a function named range() which:
  – Accepts two char parameters.
  – Prints all characters between (and including) the input two characters.
  – Print “ERROR” if the second char is < the first char.