Debugging

CS 124 – Intro to Software Development

Macbeth – Lesson 5.2

Agenda

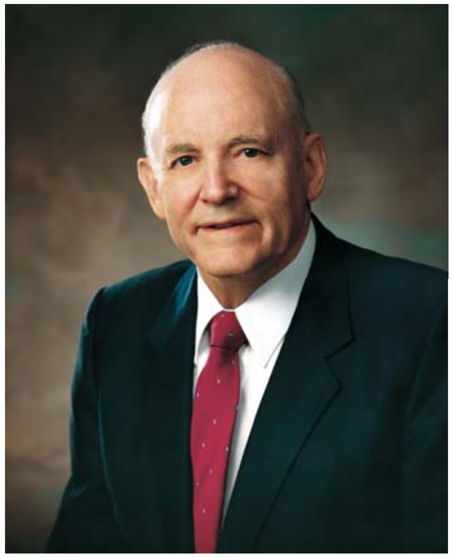
- Opening Prayer
- Spiritual Thought
- Q&A
- Debugging Techniques
    - Review Assignment
- Looking Ahead

BYU IDAHO

Spiritual Thought

Howard W. Hunter

"Let us be a temple-attending and a temple-loving people. Let us hasten to the temple as frequently as time and means and personal circumstances allow. Let us go not only for our kindred dead, but let us also go for the personal blessing of temple worship, for the sanctity and safety which is provided within those hallowed and consecrated walls. The temple is a place of beauty, it is a place of revelation, it is a place of peace. It is the house of the Lord. It is holy unto the Lord. It should be holy unto us."



Debugging Techniques

- Putting one or more `cout`'s in your code is the quick way to see what's going in your functions.
- Other techniques we have:
  - `assert's`
  - `#defines`
  - Stubs
  - Drivers

Asserts

- The `assert` command allows the programmer to verify their assumptions.
- Must add: `#include <cassert>`

```
float getDensity(float mass, float volume)
{
   assert(volume > 0);
   return (mass / volume);
}
```

- If the volume is ever 0 or negative, then the software will exit.

#defines

- The # symbol indicates a pre-processor directive for the compiler.

Directive	Description	Example
`#include`	Include a header file or a library.	`#include <iostream>`

`#define`	Creates a compiler macro. Frequently used to provide a name for literals and common expressions.	`#define PI 3.1415`
		
		`#define SQR(x) (x * x)`

`#ifdef`	Tells the compiler to use (or not use) code based on the value of a compiler macro	`#ifdef DEBUG`
		`cout << "x = " << x << endl`
		`#endif`

- Compiler macros can be set either in the code (#define) or from the command line:

  g++ -D[macro name] myfile.cpp

  g++ -DDEBUG myfile.cpp (this will set DEBUG = 1)

Stub Functions

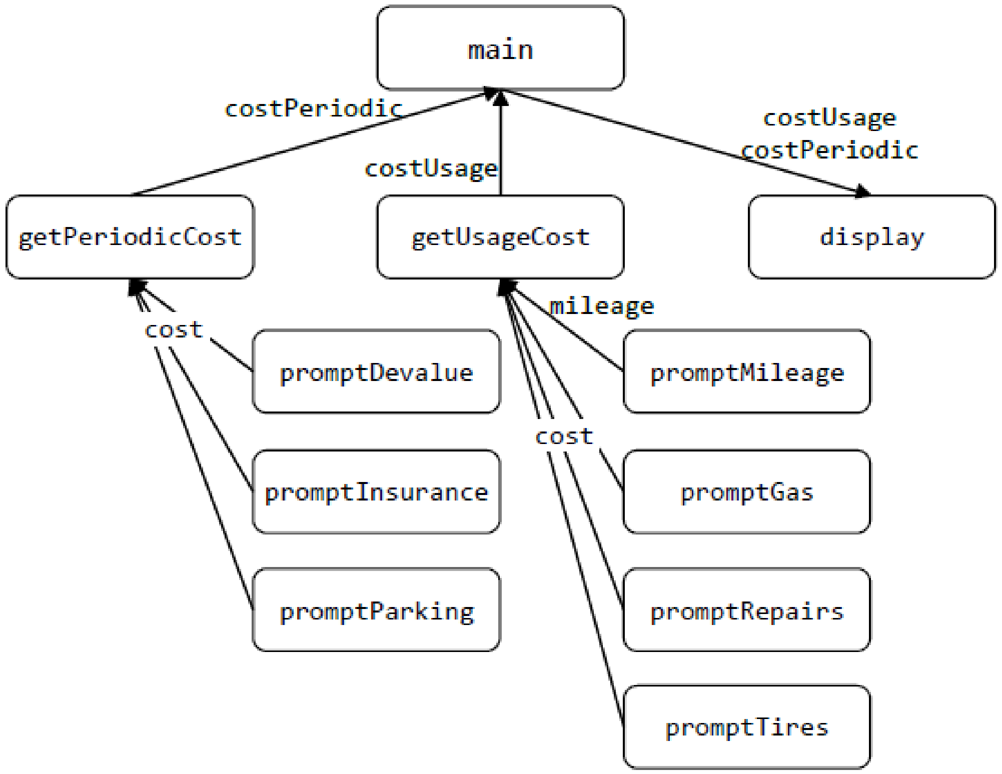
- Using the structure chart as your design, create each function.
- The only detail you put in the functions is as follows (based completely on the structure chart):
  - Function declaration including name, return type, and parameter list
  - Calls to other functions
  - Any variables and return statements necessary for it to compile
- How the function works (the implementation / algorithm) is not added at this time

```
float getIncome()
{
   return 0.0;
}

void printTithing(float income)
{
}

int main()
{
   float income = getIncome();
   printTithing(income);
   return 0;
}
```

Review Assignment



Drivers

- Remember that a function is its own mini piece of software.
- A good approach to debugging software is to debug one function at a time.
- Write code in the main function to test a single function (you can delete that code later)

```
void display(int month, int year, int offset)
{
   // Your implementation of display function
}

int main()
{
   display(10, 2017, 6);
   display(8, 2016, 3);
   display(1, 1853, 5);
   return 0;
}
```

Looking Forward

- Before Class on Friday
  - Read Section 2.2 – Designing Algorithms
  - Assignment 2.2
- Monday (not Saturday this week)
  - Project 05 is due – Please turn hardcopy with your name on it in class