“Implementation Generator”

•••

Sprint Report #2

Vincent Cote
Jace Riehl
Nathan Tipper

The Idea

- Generates a .cpp or .cc file based upon the the header file of a class
- All functions with their signatures are generated and the header file is included
- Default parameter names are given, unless they are given in the header file
- Default return values are returned from each function

Example:

```
#ifndef CALCULATOR_H
#define CALCULATOR_H


class Calculator
{
    public:
        Calculator();
        virtual ~Calculator();

        int sum(const int, const int) const;
        int mult(const int, const int) const;
        int sub(const int, const int) const;
        int div(const int, const int) const;

    protected:

    private:
};

#endif // CALCULATOR_H
```

```
Calculator::Calculator()
{
    //ctor
}

Calculator::~Calculator()
{
    //dtor
}

int Calculator::sum(const int a, const int b) const {
    return -1;
}

int Calculator::mult(const int a, const int b) const {
    return -1;
}

int Calculator::div(const int a, const int b) const {
    return -1;
}

int Calculator::sub(const int a, const int b) const {
    return -1;
}
```

Sprint #2 Plan

- Create a GUI for choosing directories
- Refactor the code
- Generate constructors and destructors based on .h
- Configure the Makefile and CI

DEMO

Sprint #2 Process

- Iterative process, trying to get used to the CodeBlocks API
- Started small, and built on that
- Scrapped the idea of “Clean Code” for “Functional Code”

Sprint #2: Things that went well

- Able to successful refactor code
- Nathan as the project lead
- Implementing the GUI

Sprint #2: Things to Improve

- Spread out the workload throughout the sprint
- Make better use of Scrum meetings.
- Configure the Makefile to work with multiple files

How these improvements will be made?

- Ask for help earlier
- Have a detailed schedule for meetings

The Next Step

- Configure the program to work with functions (return types, explicit functions)
- Implement GUI into the project
- Add a keyboard shortcut