Final Report

Overview

My autonomous code agent was created as a CLI tool that is run inside your C++ project directory. It will attempt to compile a set of C++ files and will ask ChatGPT to fix the code and provide explanations of what it changed. It is written in Rust and uses a job system written in C++ to launch specific jobs that are executed back in Rust. The three jobs that are used are a compile job, an output job, and a "fix code" job.

Features

Git Safety Check

In order to prevent accidentally overwriting code, this program will check to see if there is a git repo and if so, checks for any uncommitted changes. This will force users to save their changes so they can rollback if something does not work. This can be disabled by using `--allow-dirty`. This feature was inspired by the `cargo clippy --fix` command which will do the same thing when applying Rust's Clippy fixes.

Tweak

After getting a response from ChatGPT, the user is presented with the new code and an option to "Accept", "Tweak", or "Quit/Cancel". If tweak is chosen, the new code will be opened in the user's `$EDITOR` where they can make changes to the new code before writing it to the file. This is especially useful to get rid of extra comments or print statements generated by ChatGPT.

Warnings

If `--fix-warnings` is provided, the compiler will look for warnings and errors instead of just errors and passing both to ChatGPT.

OpenAI Token

A users OpenAI token can be provided through an environment variable, a `.env` file in the project directory, a command line argument, or a .env file in the user's `$HOME` directory.

Prompts Used

Flowscript Prompt

```
You are writing an invented programming language called flowscript that is
based on the dot language for describing graphs. Flowscript is used for
defining the order of execution with a job system with predefined job names.
Your objective is to write a job system to compile some C++ code and parse
the results. To accomplish this, you have 2 job types that you can use
"Compile" and "Output".

Flowscript begins with `diagraph {` and ends with a closing `}`.

In the middle you define both jobs and connections between them.
You must always include the job type called "input".

Other job names can be defined implicitly through connections.

For example:
`getEmail -> printEmail`
defines the "getEmail" job, the "printEmail" job and states that getEmail
must be run before printEmail.

As a whole file it would be:
digraph {
  input;
  input -> getEmail;
  getEmail -> printEmail;
}

Please give me a flowscript file that runs the "Compile" job and then the
"Output" job. Your response must start with "digraph" and end with "}". Do
not provide an explanation or any code syntax highlighting blocks.
```

The prompt worked perfectly on the first try and also for all subsequent re-prompts after that. I did a version of one-shot prompting, but instead of mimicking earlier messages with the OpenAI API, I just included the example in the singular user prompt. The only error I ran into was the Compile and Output job being lowercase, I could have fixed this by adjusting the corresponding Rust enum to be renamed by the JSON parser but simply capitalizing the words in the prompt worked perfectly fine. I made sure to tell it that other job names are defined implicitly through connections so that the final code would be shorter and hopefully have less errors. I also made sure not to tell it anything about my Flowscript syntax that would be irrelevant to the task like if statements. "Your response must start with "digraph" and end with "}"." was the most helpful part in preventing an explanation.

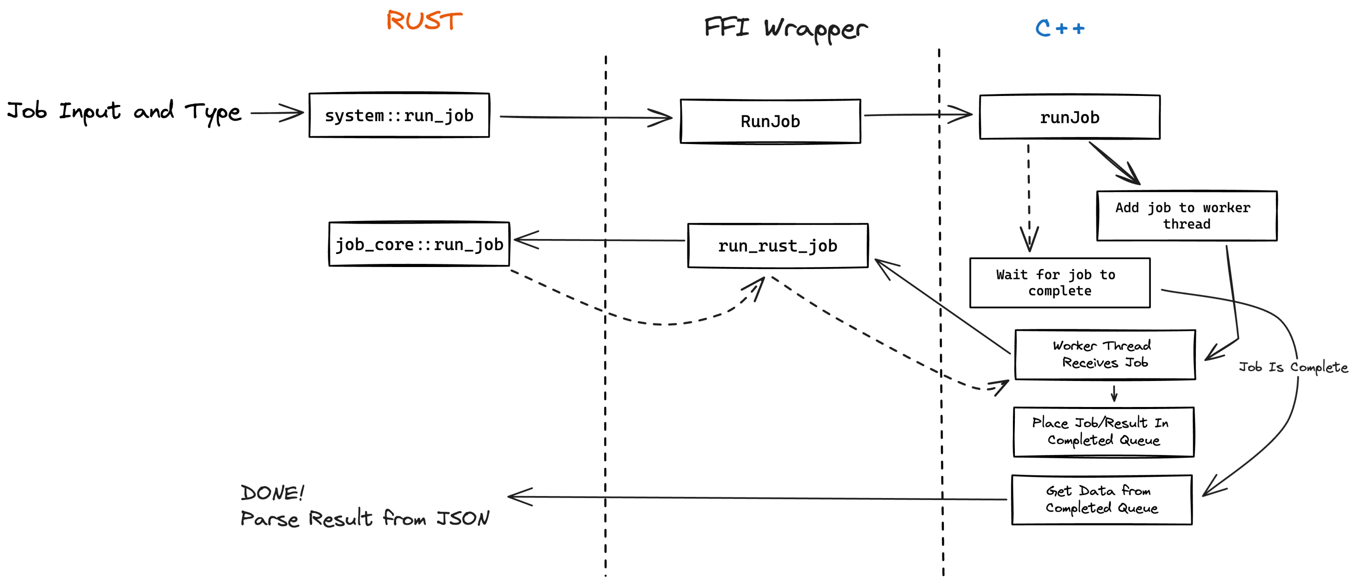
Fixing The Code

```
[
        {
                role: System,
                content: "You are an extremely smart assistant that helps with fixing c++ compiler errors  You will be given a json output of the clang compiler and then the original contents of the file. Please output a markdown response with the corrected source code and an explanation of what went wrong. Print the entire corrected source code file using the ```cpp tag, then an empty line then the explanation. You must give me the entire file, even if that means making the explanation shorter."
        },
        {
                role: User,
                content: "Compiler Output: {COMPILER JSON OUTPUT}, Original File: {ORIGINAL FILE}"
        }
]
```

I left this prompt completely unchanged from the previous lab, except for asking for the entire file. I had an issue where the LLM was not providing the entire file updated which made fixing long files hard to do, but then I created an algorithm to insert a portion of the code into the original file. However, I kept asking it to use the full file because I found I was getting better results. I did not need to do any zero/few shot prompting to get it to work.

To parse the results I just created a function that would scan the lines of the response and pull out anything contained inside triple backticks. Everything else was joined into the explanation.

Job System



The job system uses a 2-way FFI wrapper to send JSON data between the Rust program and the C++ job library. When running a job, the type of the job and it's JSON input are combined and passed to the worker thread where it will make an FFI call back to Rust with the input data and the job type and Rust will execute the job. This system made it easy for me to do the OpenAI api calls as a job, but the downside was that I had to rewrite the compiling job from the first lab. Another benefit to this system is that any struct that derives Serde's Serialize and Deserialize trait is able to be passed around in the Job system and remain strongly typed. The responses for a Job are weakly typed but still guaranteed to be valid JSON because to run a job in the system it must implement the Job trait, which I defined as:

```rust
fn run(&self) -> Result<Value>;
```

This restriction that resulted from the C++ system did not allow for any references or interior mutability within the Job structs but luckily I did not need that to create any of the jobs. It also meant that, since all of the job data was JSON strings, I had to clone a lot of data unnecessarily to move it in and out of C++ with C style string pointers.

Jobs

Compile Job

The compile job is responsible for taking a list of `PathBufs` (strings representing file locations), and compiling them together with g++. There is also an argument that determines if warnings should be looked for. The response is read from the console and parsed in to a `ClangOutputJson` data structure using Serde. Most of the fields are excluded from the compiler output since they are not needed for the LLM prompt. This is a pretty-print of the original g++ output.

```json
{
  "kind": "warning",
  "message": "#pragma once in main file",
  "children": [],
  "column-origin": 1,
  "locations": [
    {
      "caret": {
        "file": "./Code/lib/Logging.cpp",
        "line": 2,
        "display-column": 9,
        "byte-column": 9,
        "column": 9
      },
      "finish": {
        "file": "./Code/lib/Logging.cpp",
        "line": 2,
        "display-column": 12,
        "byte-column": 12,
        "column": 12
      }
    }
  ],
  "escape-source": false
}
```

This is the structure it is parsed into before it is passed to the output job:

```rust
struct ClangOutputJson {
    kind: ErrorKind,
    message: String,
    locations: Vec<LocationPair>,
}

struct LocationPair {
    caret: Location,
    finish: Option<Location>,
}

struct Location {
    file: PathBuf,
    line: i32,
    column: i32,
}
```

```
enum ErrorKind {
    Note,
    Error,
    Warning,
}
```

Output Job

The output job's job is to take a vector of `ClangOutputJson` objects and turn it into a vector of `MappedJsonError`. This makes it easier for the LLM to understand the input when there it is only given the info it needs. This is the structure of the `MappedJsonError`:

```
pub struct MappedJsonError {
    column: i32,
    line: i32,
    pub filepath: PathBuf,
    pub message: String,
    snippet: String,
}
```

For everything except `snippet`, The fields are just rearranged and accessed. There is a separate function to generate the snippet which looks at the line number and reads the file path to get the 3 lines surrounding the error.

FixCode Job

The "fix code" job takes the first element in the vector returned from the output job and provides it to an LLM (ChatGPT 4-Turbo in this case) to get back a code snippet and an explanation. The input for the job is:

```
pub struct FixCodeJob {
    pub model: Model, // ChatGpt, Mistral, or Orca
    pub output_json: MappedJsonError,
    pub file_contents: String,
}
```

and it returns:

```
pub struct FixCodeResult {
    pub code: String,
    pub explanation: String,
}
```

I use the crate `reqwest`'s blocking module to make the HTTP request to OpenAI. After the response is given, I run a function to extract out the code snippet and the explanation.

Flowscript

My flowscript library is almost completely unchanged from the previous lab except for modifying the code that calls the Job system so that I can run rust jobs. The library is accessible from one method called `execute_flowscript`. Which takes a string input of the flowscript and a `serde_json::Value` to process it with. The flowscript execution is split up into 2 parts: first it extracts the definitions of the nodes and the connections, then it builds the graph by linking the Nodes as dynamic trait objects.
This is the grammar for my flowscript. Most of the features are still available like the if statements and multithreading support.

```
WHITESPACE = _{ WHITE_SPACE }

program = {
    "digraph" ~ program_name? ~ "{" ~ program_body ~ "}" ~ ";"?
}

program_body = {
    line*
}

program_name = { (ALPHABETIC | "_")+ }

line = _{
    variable_def
  | connection_def
}

variable_def = {
    variable ~ attributes? ~ ";"
}

connection_def = {
    (variable ~ "->")+ ~ variable ~ attributes? ~ ";"
}

attributes = {
    "[" ~ attribute+ ~ "]"
}

attribute = ${
    attribute_name ~ "=" ~ quoted_attribute_value ~ " "? ~ ","?
}

quoted_attribute_value = _{ "\"" ~ attribute_value ~ "\"" }

attribute_value = @{ char* }
char = {
    !("\"" | "\\") ~ ANY
    | "\\" ~ ("\"" | "\\" | "/" | "b" | "f" | "n" | "r" | "t")
    | "\\" ~ ("u" ~ ASCII_HEX_DIGIT{4})
}

attribute_name  = ${
    ALPHABETIC+
}

variable        = ${
    ALPHABETIC+
}
```

Example Run Of Program

The following is a log of the execution for my entire "ExampleCode" folder. In my demo video I go through each folder individually but my program can also handling compiling them all at once as well.
The code in the "ExampleCode" folder is the version before any fixes have been applied.
The changes applied to the code (that we generated in this log) can be read in this commit.

Files: ["./multi-file/animal.cpp", "./multi-file/main.cpp", "./fix_warnings/main.cpp", "./simple_error/main.cpp", "./multi_errors/main.cpp", "./tweak/main.cpp"]

Just got new flowscript from chatgpt

Flowscript:

```
digraph {
  input;
  input -> Compile;
  Compile -> Output;
}
```

Starting job system....
Creating a worker thread...
I only create one worker thread because I never have anything run more than once.

Compiling: ["./multi-file/animal.cpp", "./multi-file/main.cpp", "./fix_warnings/main.cpp", "./simple_error/main.cpp", "./multi_errors/main.cpp", "./tweak/main.cpp"]

Got errors:

```
 [
  {
    "column": 5,
    "line": 6,
    "filepath": "./multi-file/animal.cpp",
    "message": "'print' was not declared in this scope; did you mean
'printf'?",
    "snippet": "public:\n  void speak() {\n    print(\"hey im a animal\");\n
}\n};\n\n"
  },
  {
    "column": 5,
    "line": 13,
    "filepath": "./multi-file/animal.cpp",
    "message": "'print' was not declared in this scope; did you mean
'printf'?",
    "snippet": "public:\n  void speak() {\n    print(\"hey im a dog\");\n
};\n};\n\n"
  },
  {
    "column": 5,
    "line": 20,
    "filepath": "./multi-file/animal.cpp",
    "message": "'print' was not declared in this scope; did you mean
'printf'?",
    "snippet": "public:\n  void speak() {\n    print(\"hey im a cat\");\n
}\n};\n"
  },
  {
    "column": 5,
    "line": 6,
    "filepath": "./multi-file/./animal.cpp",
    "message": "'print' was not declared in this scope; did you mean
'printf'?",
    "snippet": "public:\n  void speak() {\n    print(\"hey im a animal\");\n
}\n};\n\n"
  },
  {
    "column": 5,
    "line": 13,
    "filepath": "./multi-file/./animal.cpp",
    "message": "'print' was not declared in this scope; did you mean
'printf'?",
    "snippet": "public:\n  void speak() {\n    print(\"hey im a dog\");\n
};\n};\n\n"
  },
  {
    "column": 5,
    "line": 20,
    "filepath": "./multi-file/./animal.cpp",
    "message": "'print' was not declared in this scope; did you mean
'printf'?",
    "snippet": "public:\n  void speak() {\n    print(\"hey im a cat\");\n
}\n};\n"
  },
  {
    "column": 10,
    "line": 9,
    "filepath": "./multi-file/main.cpp",
    "message": "'class Dog' has no member named 'speek'; did you mean
'speak'?",
    "snippet": "    // Speak twice\n    dog->speak();\n    dog->speek();\n\n
return 0;\n}\n"
  },
  {
    "column": 10,
    "line": 2,
    "filepath": "./simple_error/main.cpp",
    "message": "'cout' is not a member of 'std'",
    "snippet": "int main() {\n    std::cout << \"Hello World!\";\n    return
0;\n}\n"
  },
  {
    "column": 10,
    "line": 11,
    "filepath": "./multi_errors/main.cpp",
    "message": "'vector' is not a member of 'std'",
    "snippet": "\nint main() {\n    std::vector<Contact> contacts;\n
while (true) {\n        Contact newContact\n        std::cout << \"Enter
contact name (or 'exit' to finish): \";\n"
  },
  {
    "column": 24,
    "line": 11,
    "filepath": "./multi_errors/main.cpp",
    "message": "expected primary-expression before '>' token",
    "snippet": "\nint main() {\n    std::vector<Contact> contacts;\n
while (true) {\n        Contact newContact\n        std::cout << \"Enter
contact name (or 'exit' to finish): \";\n"
  },
  {
    "column": 26,
    "line": 11,
    "filepath": "./multi_errors/main.cpp",
    "message": "'contacts' was not declared in this scope; did you mean
'Contact'?",
    "snippet": "\nint main() {\n    std::vector<Contact> contacts;\n
while (true) {\n        Contact newContact\n        std::cout << \"Enter
contact name (or 'exit' to finish): \";\n"
  },
  {
    "column": 9,
    "line": 14,
    "filepath": "./multi_errors/main.cpp",
    "message": "expected initializer before 'std'",
    "snippet": "    while (true) {\n        Contact newContact\n
std::cout << \"Enter contact name (or 'exit' to finish): \";\n
std::getline(std::cin, newContact.name);\n        if (newContact.name ==
\"exit\") {\n            break;\n"
  },
  {
    "column": 32,
    "line": 15,
    "filepath": "./multi_errors/main.cpp",
    "message": "'newContact' was not declared in this scope; did you mean
'Contact'?",
    "snippet": "        Contact newContact\n        std::cout << \"Enter
contact name (or 'exit' to finish): \";\n        std::getline(std::cin,
newContact.name);\n        if (newContact.name == \"exit\") {\n
break;\n        }\n"
  },
  {
    "column": 5,
    "line": 4,
    "filepath": "./tweak/main.cpp",
    "message": "'seifjosijeofj238h' was not declared in this scope",
    "snippet": "\nint main() {\n    seifjosijeofj238h\n    return 0;\n}\n"
  }
]
```

Asking ChatGPT to fix first error in the list using input:

FixCodeJob:

```
{
  "model": "ChatGpt",
  "output_json": {
    "column": 5,
    "line": 6,
    "filepath": "./multi-file/animal.cpp",
    "message": "'print' was not declared in this scope; did you mean
'printf'?",
    "snippet": "public:\n  void speak() {\n    print(\"hey im a animal\");\n
}\n};\n\n"
  },
  "file_contents": "#include <iostream>\n\nclass Animal {\npublic:\n  void
speak() {\n    print(\"hey im a animal\");\n  }\n};\n\nclass Dog : public
Animal {\npublic:\n  void speak() {\n    print(\"hey im a dog\");\n
};\n};\n\nclass Cat : public Animal {\npublic:\n  void speak() {\n
print(\"hey im a cat\");\n  }\n};\n"
}
```

FixCodeResult:

```
{
  "code": "#include <iostream>\n\nclass Animal {\npublic:\n  void speak()
{\n    std::cout << \"hey im a animal\" << std::endl;\n  }\n};\n\nclass Dog
: public Animal {\npublic:\n  void speak() {\n    std::cout << \"hey im a
dog\" << std::endl;\n  }\n};\n\nclass Cat : public Animal {\npublic:\n  void
speak() {\n    std::cout << \"hey im a cat\" << std::endl;\n  }\n};\n",
  "explanation": "\nThe error message indicates that the function `print`
was not declared in the supplied scope. In C++, the standard print function
is called `std::cout`, which is used for stream output and must be used with
the insertion operator (`<<`). The call to `print` has been replaced with
`std::cout << \"message\" << std::endl;` where `std::endl` is used for
adding a newline and flushing the stream. The `std::` prefix is necessary to
indicate that `cout` and `endl` are being used from the standard
namespace.\n"
}
```

ChatGPT answered correctly, no need to tweak.

Compiling Files: ["./multi-file/animal.cpp", "./multi-file/main.cpp", "./fix_warnings/main.cpp", "./simple_error/main.cpp", "./multi_errors/main.cpp", "./tweak/main.cpp"]

Got errors:

```
[
 {
   "column": 10,
   "line": 9,
   "filepath": "./multi-file/main.cpp",
   "message": "'class Dog' has no member named 'speek'; did you mean
'speak'?",
   "snippet": "    // Speak twice\n    dog->speak();\n    dog->speek();\n\n
return 0;\n}\n"
 },
 {
   "column": 10,
   "line": 2,
   "filepath": "./simple_error/main.cpp",
   "message": "'cout' is not a member of 'std'",
   "snippet": "int main() {\n    std::cout << \"Hello World!\";\n    return
0;\n}\n"
 },
 {
   "column": 10,
   "line": 11,
   "filepath": "./multi_errors/main.cpp",
   "message": "'vector' is not a member of 'std'",
   "snippet": "\nint main() {\n    std::vector<Contact> contacts;\n    while
(true) {\n        Contact newContact\n        std::cout << \"Enter contact
name (or 'exit' to finish): \";\n"
 },
 {
   "column": 24,
   "line": 11,
   "filepath": "./multi_errors/main.cpp",
   "message": "expected primary-expression before '>' token",
   "snippet": "\nint main() {\n    std::vector<Contact> contacts;\n    while
(true) {\n        Contact newContact\n        std::cout << \"Enter contact
name (or 'exit' to finish): \";\n"
 },
 {
   "column": 26,
   "line": 11,
   "filepath": "./multi_errors/main.cpp",
   "message": "'contacts' was not declared in this scope; did you mean
'Contact'?",
   "snippet": "\nint main() {\n    std::vector<Contact> contacts;\n    while
(true) {\n        Contact newContact\n        std::cout << \"Enter contact
```

```
name (or 'exit' to finish): \";\n"
 },
 {
   "column": 9,
   "line": 14,
   "filepath": "./multi_errors/main.cpp",
   "message": "expected initializer before 'std'",
   "snippet": "    while (true) {\n        Contact newContact\n
std::cout << \"Enter contact name (or 'exit' to finish): \";\n
std::getline(std::cin, newContact.name);\n        if (newContact.name ==
\"exit\") {\n            break;\n"
 },
 {
   "column": 32,
   "line": 15,
   "filepath": "./multi_errors/main.cpp",
   "message": "'newContact' was not declared in this scope; did you mean
'Contact'?",
   "snippet": "        Contact newContact\n        std::cout << \"Enter
contact name (or 'exit' to finish): \";\n        std::getline(std::cin,
newContact.name);\n        if (newContact.name == \"exit\") {\n
break;\n        }\n"
 },
 {
   "column": 5,
   "line": 4,
   "filepath": "./tweak/main.cpp",
   "message": "'seifjosijeofj238h' was not declared in this scope",
   "snippet": "\nint main() {\n    seifjosijeofj238h\n    return 0;\n}\n"
 }
]
```

Asking ChatGPT to fix first error using input:

FixCodeJob:

```
{
  "model": "ChatGpt",
  "output_json": {
    "column": 10,
    "line": 9,
    "filepath": "./multi-file/main.cpp",
    "message": "'class Dog' has no member named 'speek'; did you mean
'speak'?",
    "snippet": "    // Speak twice\n    dog->speak();\n    dog->speek();\n\n
return 0;\n}\n"
```

```
  },
  "file_contents": "#include <iostream>\n#include \"./animal.cpp\"\n\nint main() {\n    Dog *dog = new Dog();\n\n    // Speak twice\n    dog->speak();\n    dog->speek();\n\n    return 0;\n}\n"
}
```

FixCodeResult:

```
{
  "code": "#include <iostream>\n#include \"./animal.cpp\"\n\nint main() {\n    Dog *dog = new Dog();\n\n    // Speak twice\n    dog->speak();\n    dog->speak(); // Corrected from speek() to speak()\n\n    return 0;\n}\n",
  "explanation": "\nThe compiler error indicates that in the `main.cpp` file, within the `main` function, there is a call to a member function named `speek()` of the `Dog` class which does not exist. It suggests that maybe the correct member function name is `speak()`. \n\nThe correction is to change `dog->speek();` to `dog->speak();`. This fixes the typo in the method call, adhering to the actual declaration of the `speak` method within the `Dog` class.\n"
}
```

ChatGPT answered correctly, no need to tweak. Pretty easy for it to answer

Compiling: ["./multi-file/animal.cpp", "./multi-file/main.cpp", "./fix_warnings/main.cpp", "./simple_error/main.cpp", "./multi_errors/main.cpp", "./tweak/main.cpp"]

Got errors:

```
 [
  {
    "column": 10,
    "line": 2,
    "filepath": "./simple_error/main.cpp",
    "message": "'cout' is not a member of 'std'",
    "snippet": "int main() {\n    std::cout << \"Hello World!\";\n    return 0;\n}\n"
  },
  {
    "column": 10,
    "line": 11,
    "filepath": "./multi_errors/main.cpp",
    "message": "'vector' is not a member of 'std'",
    "snippet": "\nint main() {\n    std::vector<Contact> contacts;\n    while (true) {\n        Contact newContact\n        std::cout << \"Enter contact name (or 'exit' to finish): \";\n"
  },
  {
    "column": 24,
    "line": 11,
    "filepath": "./multi_errors/main.cpp",
    "message": "expected primary-expression before '>' token",
    "snippet": "\nint main() {\n    std::vector<Contact> contacts;\n
while (true) {\n        Contact newContact\n        std::cout << \"Enter
contact name (or 'exit' to finish): \";\n"
  },
  {
    "column": 26,
    "line": 11,
    "filepath": "./multi_errors/main.cpp",
    "message": "'contacts' was not declared in this scope; did you mean
'Contact'?",
    "snippet": "\nint main() {\n    std::vector<Contact> contacts;\n
while (true) {\n        Contact newContact\n        std::cout << \"Enter
contact name (or 'exit' to finish): \";\n"
  },
  {
    "column": 9,
    "line": 14,
    "filepath": "./multi_errors/main.cpp",
    "message": "expected initializer before 'std'",
    "snippet": "    while (true) {\n        Contact newContact\n
std::cout << \"Enter contact name (or 'exit' to finish): \";\n
std::getline(std::cin, newContact.name);\n        if (newContact.name ==
\"exit\") {\n            break;\n"
  },
  {
    "column": 32,
    "line": 15,
    "filepath": "./multi_errors/main.cpp",
    "message": "'newContact' was not declared in this scope; did you mean
'Contact'?",
    "snippet": "        Contact newContact\n        std::cout << \"Enter
contact name (or 'exit' to finish): \";\n        std::getline(std::cin,
newContact.name);\n        if (newContact.name == \"exit\") {\n
break;\n        }\n"
  },
  {
    "column": 5,
    "line": 4,
    "filepath": "./tweak/main.cpp",
    "message": "'seifjosijeofj238h' was not declared in this scope",
    "snippet": "\nint main() {\n    seifjosijeofj238h\n    return 0;\n}\n"
  }
]
```

Asking ChatGPT to fix first error using input:

FixCodeJob:

```
{
  "model": "ChatGpt",
  "output_json": {
    "column": 10,
    "line": 2,
    "filepath": "./simple_error/main.cpp",
    "message": "'cout' is not a member of 'std'",
    "snippet": "int main() {\n    std::cout << \"Hello World!\";\n    return 0;\n}\n"
  },
  "file_contents": "int main() {\n    std::cout << \"Hello World!\";\n    return 0;\n}\n"
}
```

FixCodeResult:

```
{
  "code": "#include <iostream>\n\nint main() {\n    std::cout << \"Hello World!\";\n    return 0;\n}\n",
  "explanation": "\nThe error message `\"'cout' is not a member of 'std'\"` indicates that the `cout` object, which resides in the `std` namespace, is not recognized. This usually happens when the `iostream` header, which declares `std::cout`, is not included at the beginning of the file. Adding `#include <iostream>` at the top of the file resolves the issue.\n"
}
```

ChatGPT nailed it again, no need to tweak. This file is done but will continue getting compiled.

Compiling: ["./multi-file/animal.cpp", "./multi-file/main.cpp", "./fix_warnings/main.cpp", "./simple_error/main.cpp", "./multi_errors/main.cpp", "./tweak/main.cpp"]

Got errors:

```
 [
  {
    "column": 10,
    "line": 11,
    "filepath": "./multi_errors/main.cpp",
    "message": "'vector' is not a member of 'std'",
    "snippet": "\nint main() {\n    std::vector<Contact> contacts;\n
while (true) {\n        Contact newContact\n        std::cout << \"Enter
contact name (or 'exit' to finish): \";\n"
  },
  {
    "column": 24,
    "line": 11,
    "filepath": "./multi_errors/main.cpp",
    "message": "expected primary-expression before '>' token",
    "snippet": "\nint main() {\n    std::vector<Contact> contacts;\n
while (true) {\n        Contact newContact\n        std::cout << \"Enter
contact name (or 'exit' to finish): \";\n"
  },
  {
    "column": 26,
    "line": 11,
    "filepath": "./multi_errors/main.cpp",
    "message": "'contacts' was not declared in this scope; did you mean
'Contact'?",
    "snippet": "\nint main() {\n    std::vector<Contact> contacts;\n
while (true) {\n        Contact newContact\n        std::cout << \"Enter
contact name (or 'exit' to finish): \";\n"
  },
  {
    "column": 9,
    "line": 14,
    "filepath": "./multi_errors/main.cpp",
    "message": "expected initializer before 'std'",
    "snippet": "    while (true) {\n        Contact newContact\n
std::cout << \"Enter contact name (or 'exit' to finish): \";\n
std::getline(std::cin, newContact.name);\n        if (newContact.name ==
\"exit\") {\n            break;\n"
  },
  {
    "column": 32,
    "line": 15,
    "filepath": "./multi_errors/main.cpp",
    "message": "'newContact' was not declared in this scope; did you mean
'Contact'?",
    "snippet": "        Contact newContact\n        std::cout << \"Enter
contact name (or 'exit' to finish): \";\n        std::getline(std::cin,
newContact.name);\n        if (newContact.name == \"exit\") {\n
break;\n        }\n"
```

```
  },
  {
    "column": 5,
    "line": 4,
    "filepath": "./tweak/main.cpp",
    "message": "'seifjosijeofj238h' was not declared in this scope",
    "snippet": "\nint main() {\n    seifjosijeofj238h\n    return 0;\n}\n"
  }
]
```

Asking ChatGPT to fix first error using input:

FixCodeJob:

```
{
  "model": "ChatGpt",
  "output_json": {
    "column": 10,
    "line": 11,
    "filepath": "./multi_errors/main.cpp",
    "message": "'vector' is not a member of 'std'",
    "snippet": "\nint main() {\n    std::vector<Contact> contacts;\n
while (true) {\n        Contact newContact\n        std::cout << \"Enter
contact name (or 'exit' to finish): \";\n"
  },
  "file_contents": "#include <iostream>\n\nstruct Contact {\n    std::string
name;\n    std::string address;\n    std::string phoneNumber;\n    Contact*
knows; // Pointer to another contact that this contact knows\n};\n\nint
main() {\n    std::vector<Contact> contacts;\n    while (true) {\n
Contact newContact\n        std::cout << \"Enter contact name (or 'exit' to
finish): \";\n        std::getline(std::cin, newContact.name);\n        if
(newContact.name == \"exit\") {\n            break;\n        }\n
std::cout << \"Enter contact address: \";\n        std::getline(std::cin,
newContact.address);\n        std::cout << \"Enter contact phone number:
\";\n        std::getline(std::cin, newContact.phoneNumber);\n\n
std::string knowsName;\n        std::cout << \"Enter the name of someone
this contact knows (or 'none'): \";\n        std::getline(std::cin,
knowsName);\n        if (knowsName != \"none\") {\n            for (auto
contact : contacts) {\n                if (contact.name == knowsName) {\n
newContact.knows = &contact;\n                    break;\n
}\n            }\n        } else {\n            newContact.kows = nullptr;\n
}\n\n        contacts.push_back(newContact);\n    }\n\n    std::cout <<
\"\\nContact List:\\n\";\n    for (const auto& contact : contacts) {\n
std::cout << \"Name: \" << contact.name << \"\\nAddress: \" <<
contact.address\n                  << \"\\nPhone Number: \" <<
contact.phoneNumber;\n        if (contact.knows != nullptr) {\n
std::cout << \"\\nKnows: \" << contact.knows->name;\n        }\n
std::cout << \"\\n\\n\";\n    }\n\n    return 0;\n}\n"
}
```

FixCodeResult:

```
{
  "code": "#include <iostream>\n#include <vector> // Include vector
header\n\nstruct Contact {\n    std::string name;\n    std::string
address;\n    std::string phoneNumber;\n    Contact* knows; // Pointer to
another contact that this contact knows\n};\n\nint main() {\n
std::vector<Contact> contacts;\n    while (true) {\n        Contact
newContact; // Add a semicolon here\n        std::cout << \"Enter contact
name (or 'exit' to finish): \";\n        std::getline(std::cin,
newContact.name);\n        if (newContact.name == \"exit\") {\n
break;\n        }\n        std::cout << \"Enter contact address: \";\n
std::getline(std::cin, newContact.address);\n        std::cout << \"Enter
contact phone number: \";\n        std::getline(std::cin,
newContact.phoneNumber);\n\n        std::string knowsName;\n
std::cout << \"Enter the name of someone this contact knows (or 'none'):
\";\n        std::getline(std::cin, knowsName);\n        if (knowsName !=
\"none\") {\n            for (auto& contact : contacts) { // Changed from
auto to auto& to be able to take the address properly\n                if
(contact.name == knowsName) {\n                    newContact.knows =
&contact;\n                    break;\n                }\n            }\n
} else {\n            newContact.knows = nullptr; // Fix typo from 'kows' to
'knows'\n        }\n\n        contacts.push_back(newContact);\n    }\n\n
std::cout << \"\\nContact List:\\n\";\n    for (const auto& contact :
contacts) {\n        std::cout << \"Name: \" << contact.name <<
\"\\nAddress: \" << contact.address\n                  << \"\\nPhone Number:
\" << contact.phoneNumber;\n        if (contact.knows != nullptr) {\n
std::cout << \"\\nKnows: \" << contact.knows->name;\n        }\n
std::cout << \"\\n\\n\";\n    }\n\n    return 0;\n}\n",
  "explanation": "\nExplanation:\n\nThe compiler error indicates that vector
is not a member of the std namespace. This usually occurs because the vector
header has not been included. To fix this, we need to include the `<vector>`
header at the beginning of the file.\n\nThere was also a missing semicolon
after the declaration of newContact which would cause a compile error.\n\nTo
assign the address of a contact in the vector to `newContact.knows`, the
loop should use `auto&` instead of `auto` to avoid copying the contacts and
to obtain the correct reference to use as an address. \n\nLastly, there was
a typo in setting `newContact.knows` to `nullptr`. The code mistakenly used
`newContact.kows` which would not compile. This was corrected to
`newContact.knows`.\n"
}
```

ChatGPT was only passed one error but fixed every error in the file, suggesting that for lots of issues, I don't even need to compile, I just need to pass the source file if there is an error and it will "test compile" itself.

Compiling: ["./multi-file/animal.cpp", "./multi-file/main.cpp", "./fix_warnings/main.cpp", "./simple_error/main.cpp", "./multi_errors/main.cpp", "./tweak/main.cpp"]

Got errors:

```
 [
  {
    "column": 5,
    "line": 4,
    "filepath": "./tweak/main.cpp",
    "message": "'seifjosijeofj238h' was not declared in this scope",
    "snippet": "\nint main() {\n    seifjosijeofj238h\n    return 0;\n}\n"
  }
]
```

Asking ChatGPT to fix first error using input:

FixCodeJob:

```
{
  "model": "ChatGpt",
  "output_json": {
    "column": 5,
    "line": 4,
    "filepath": "./tweak/main.cpp",
    "message": "'seifjosijeofj238h' was not declared in this scope",
    "snippet": "\nint main() {\n    seifjosijeofj238h\n    return 0;\n}\n"
  },
  "file_contents": "#include <iostream>\n\nint main() {\n    seifjosijeofj238h\n    return 0;\n}\n"
}
```

FixCodeResult:

```
{
  "code": "#include <iostream>\n\nint main() {\n    // seifjosijeofj238h // This line seems to be errant or a placeholder\n    std::cout << \"Hello, World!\" << std::endl; // Corrected line to print \"Hello, World!\" to the console\n    return 0;\n}\n",
  "explanation": "\nThe error message from the compiler indicated that
`seifjosijeofj238h` was not declared in the scope of `main`. This suggests
that `seifjosijeofj238h` was intended to be either a variable, a function,
or some other identifiable entity in C++, but since there's no definition,
declaration, or relevant functionality associated with it, this seems to be
a line of nonsensical text or a placeholder. This is not valid C++ syntax
and needs to be either defined, declared, or removed.\n\nIn the corrected
code, I have commented out this errant line and added a line that uses
`std::cout` from the `iostream` header to print the standard \"Hello,
World!\" message to the console. If `seifjosijeofj238h` was intended to be
something specific, you would need to replace that with the appropriate
content.\n"
}
```

Leaves a lot of comments so I did a tweak to remove them.

Recompiling:...
No more errors!!!

Destroying job system...