First VPLanet Developers Workshop

Lesson 5
How to Add an Output

Overview

Outputs are easier than options, so we start here (no verify step)

Four steps:

- Determine which module(s) the output applies to
- Define a new integer ID
- Add text to InitializeOutput
- Create Write function

To add to repo:

- Add new test
- Run Valgrind
- Update documentation (if necessary)
- Update examples (if necessary)
- Issue Pull Request

Worked Example: Adding Pericenter Distance

Pericenter is the point of closest approach of an orbit to the central mass

= a(1-e)

= SemiMajorAxis*(1 - Eccentricity)

= SemiMajorAxis*(1 - sqrt(Hecc*Hecc + Kecc*Kecc)

What the heck is Hecc?

- It's a change of variable from eccentricity and longitude of pericenter
- Hecc = Eccentricity * sin(long. of peri.)
- Kecc = Eccentricity * cos(long. of peri.)
- As e -> 0, long of peri becomes ill-defined; Hecc and Kecc don't
- Hecc and Kecc sometimes called Poincaré variables
- Hecc and Kecc are the Primary Variables for DistOrb

(Note that obliquity and precession angle similarly transformed)

Step 1: Which Module(s)?

Pericenter is an orbital property

- SpiNBody
- DistOrb
- EqTide
- BINARY (we'll ignore this one for now)
- GalHabit (we'll ignore this one for now)

So pericenter is "multi-module", we need to put it in output.[ch]

If it only applied to a single module, we'd put it in those source files

Step 2: Define an Integer ID

Each output is defined a unique integer identifier

Each module has a unique range

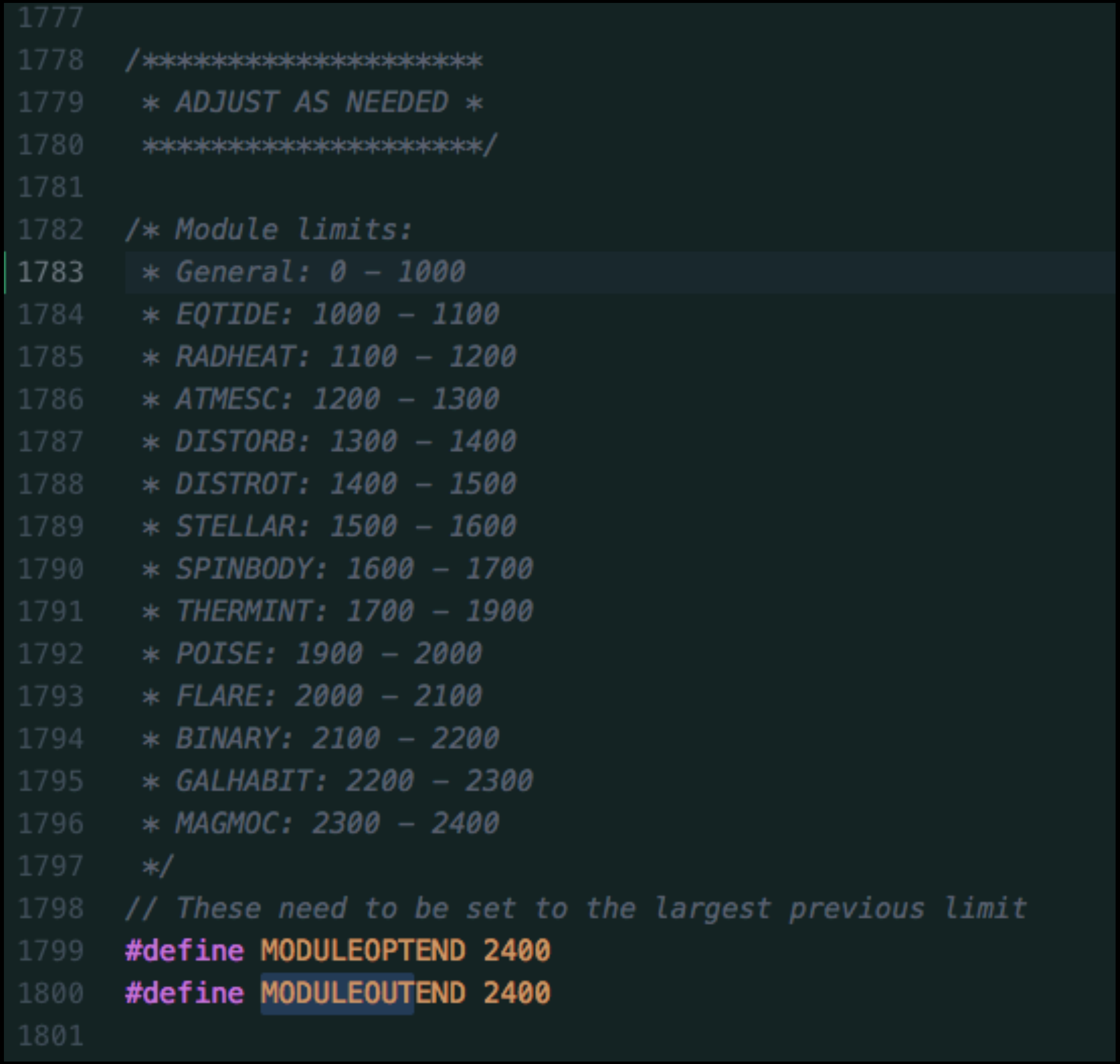
Multi-module, or general, outputs also have a range

These are defined in vplanet.h

Pericenter is general, so we will need to pick a value in its range

Step 2: Define an Integer ID

vplanet.h



```
/********************
 * ADJUST AS NEEDED *
 ********************/

/* Module limits:
 * General: 0 - 1000
 * EQTIDE: 1000 - 1100
 * RADHEAT: 1100 - 1200
 * ATMESC: 1200 - 1300
 * DISTORB: 1300 - 1400
 * DISTROT: 1400 - 1500
 * STELLAR: 1500 - 1600
 * SPINBODY: 1600 - 1700
 * THERMINT: 1700 - 1900
 * POISE: 1900 - 2000
 * FLARE: 2000 - 2100
 * BINARY: 2100 - 2200
 * GALHABIT: 2200 - 2300
 * MAGMOC: 2300 - 2400
 */
// These need to be set to the largest previous limit
#define MODULEOPTEND 2400
#define MODULEOUTEND 2400
```

General outputs are 0 — 999

Other outputs will be in the appropriate range

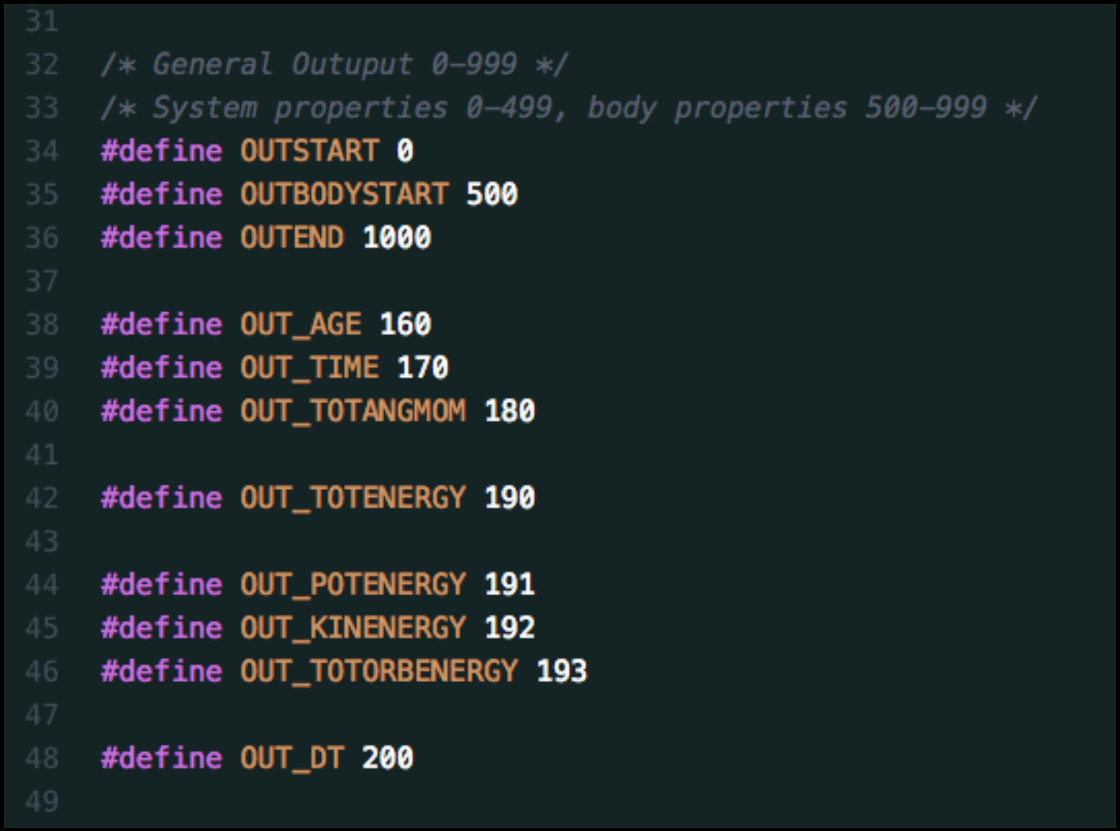
(Note if you add a new module (Lesson 9), you will need to update MODULEOPTEND and MODULEOUTEND)

Step 2: Define an Integer ID

OK, so pericenter is 0 - 999, but what should it be?

We need to look in output.c to find an available integer

output.h



```
/* General Outuput 0-999 */
/* System properties 0-499, body properties 500-999 */
#define OUTSTART 0
#define OUTBODYSTART 500
#define OUTEND 1000

#define OUT_AGE 160
#define OUT_TIME 170
#define OUT_TOTANGMOM 180

#define OUT_TOTENERGY 190

#define OUT_POTENERGY 191
#define OUT_KINENERGY 192
#define OUT_TOTORBENERGY 193

#define OUT_DT 200

```

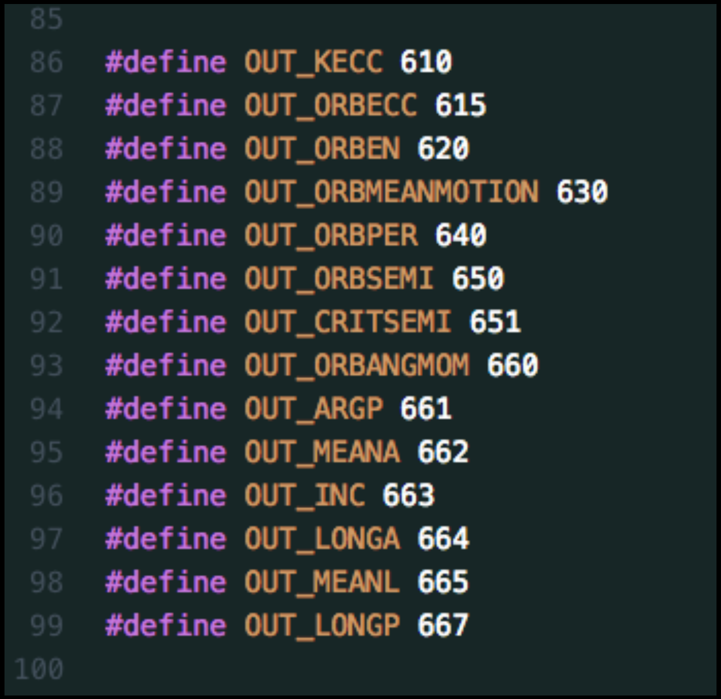
General outputs are divided into system-wide and body outputs

Pericenter is unique for each body, so it must be 500-999

Step 2: Define an Integer ID

OK, so what value between 500 and 999?

Try to pick a number that makes sense, given what has been taken



```
#define OUT_KECC 610
#define OUT_ORBECC 615
#define OUT_ORBEN 620
#define OUT_ORBMEANMOTION 630
#define OUT_ORBPER 640
#define OUT_ORBSEMI 650
#define OUT_CRITSEMI 651
#define OUT_ORBANGMOM 660
#define OUT_ARGP 661
#define OUT_MEANA 662
#define OUT_INC 663
#define OUT_LONGA 664
#define OUT_MEANL 665
#define OUT_LONGP 667
```

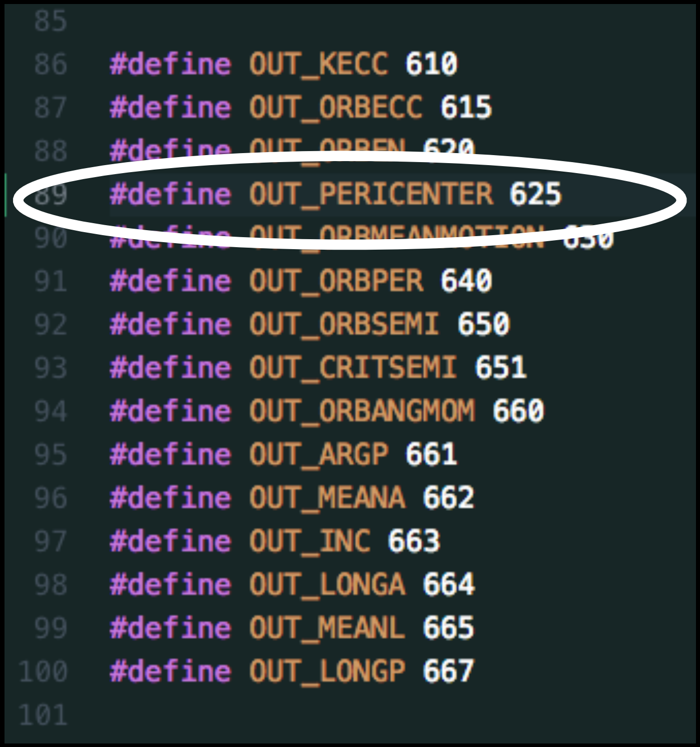
Here's a block of numbers that looks related to orbits, let's pick 625

I try to leave room for new outputs to be added

Step 2: Define an Integer ID

OK, so what value between 500 and 999?

Try to pick a number that makes sense, given what has been taken



```
#define OUT_KECC 610
#define OUT_ORBECC 615
#define OUT_ORBEN 620
#define OUT_PERICENTER 625
#define OUT_ORBMEANMOTION 630
#define OUT_ORBPER 640
#define OUT_ORBSEMI 650
#define OUT_CRITSEMI 651
#define OUT_ORBANGMOM 660
#define OUT_ARGP 661
#define OUT_MEANA 662
#define OUT_INC 663
#define OUT_LONGA 664
#define OUT_MEANL 665
#define OUT_LONGP 667
```

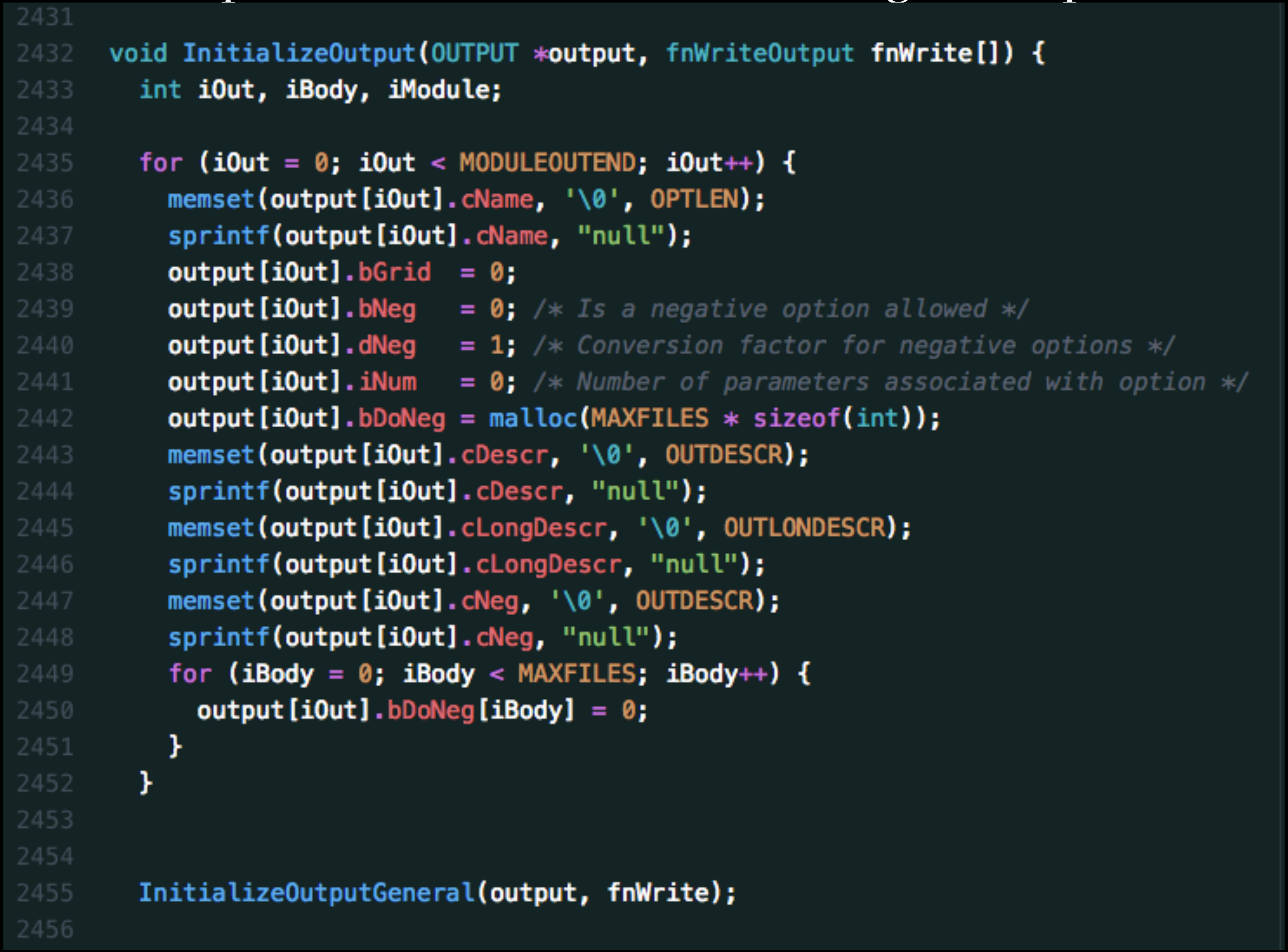
Here’s a block of numbers that looks related to orbits, let’s pick 625

I try to leave room for new outputs to be added

Step 3: Update InitializeOutput

Next we need to let VPLanet know how to implement our new output

The InitializeOutput functions do that (starting in output.c)

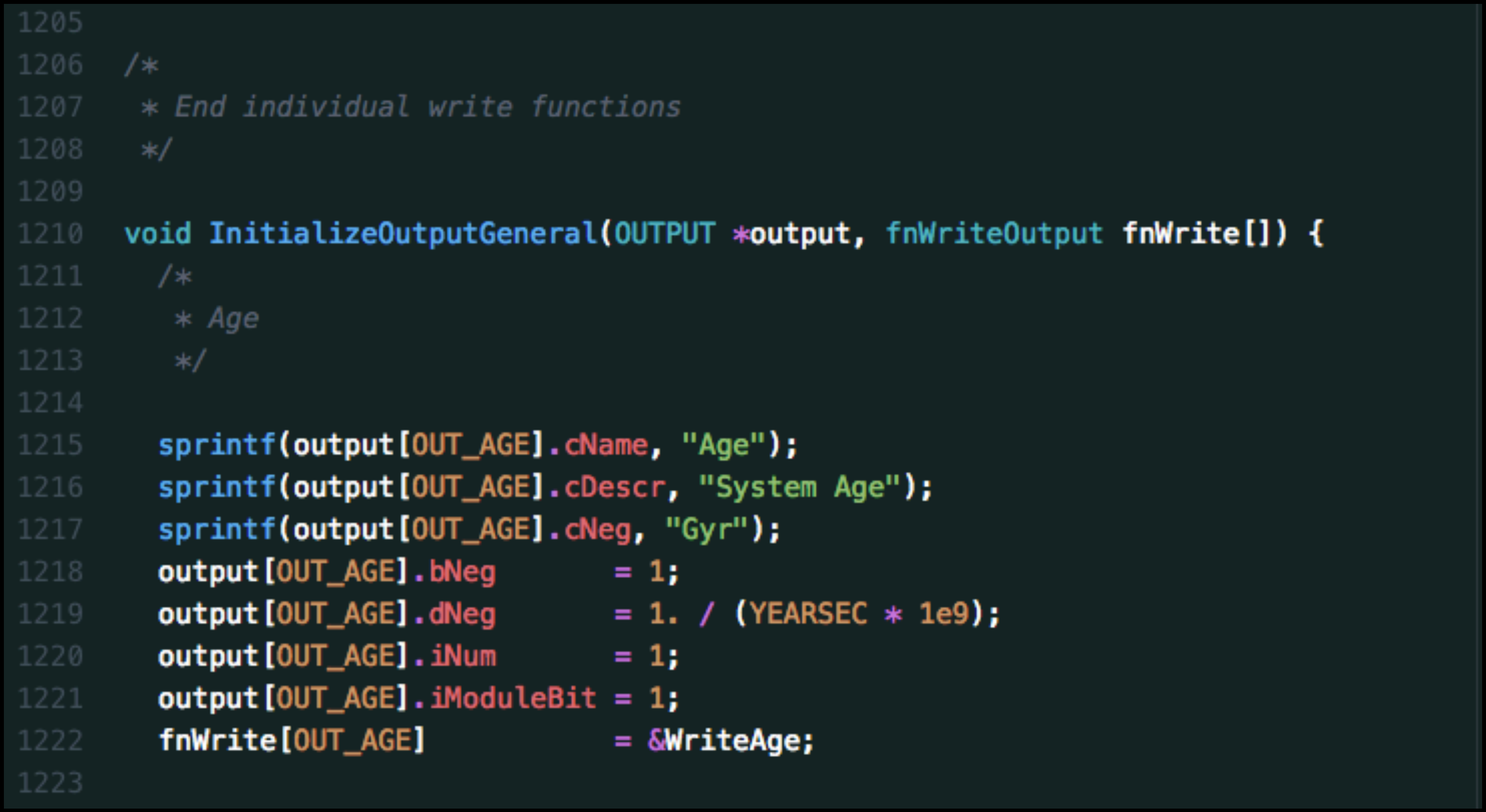


```

void InitializeOutput(OUTPUT *output, fnWriteOutput fnWrite[]) {
  int iOut, iBody, iModule;

  for (iOut = 0; iOut < MODULEOUTEND; iOut++) {
    memset(output[iOut].cName, '\0', OPTLEN);
    sprintf(output[iOut].cName, "null");
    output[iOut].bGrid  = 0;
    output[iOut].bNeg   = 0; /* Is a negative option allowed */
    output[iOut].dNeg   = 1; /* Conversion factor for negative options */
    output[iOut].iNum   = 0; /* Number of parameters associated with option */
    output[iOut].bDoNeg = malloc(MAXFILES * sizeof(int));
    memset(output[iOut].cDescr, '\0', OUTDESCR);
    sprintf(output[iOut].cDescr, "null");
    memset(output[iOut].cLongDescr, '\0', OUTLONDESCR);
    sprintf(output[iOut].cLongDescr, "null");
    memset(output[iOut].cNeg, '\0', OUTDESCR);
    sprintf(output[iOut].cNeg, "null");
    for (iBody = 0; iBody < MAXFILES; iBody++) {
      output[iOut].bDoNeg[iBody] = 0;
    }
  }


  InitializeOutputGeneral(output, fnWrite);

```

Step 3: Update InitializeOutput

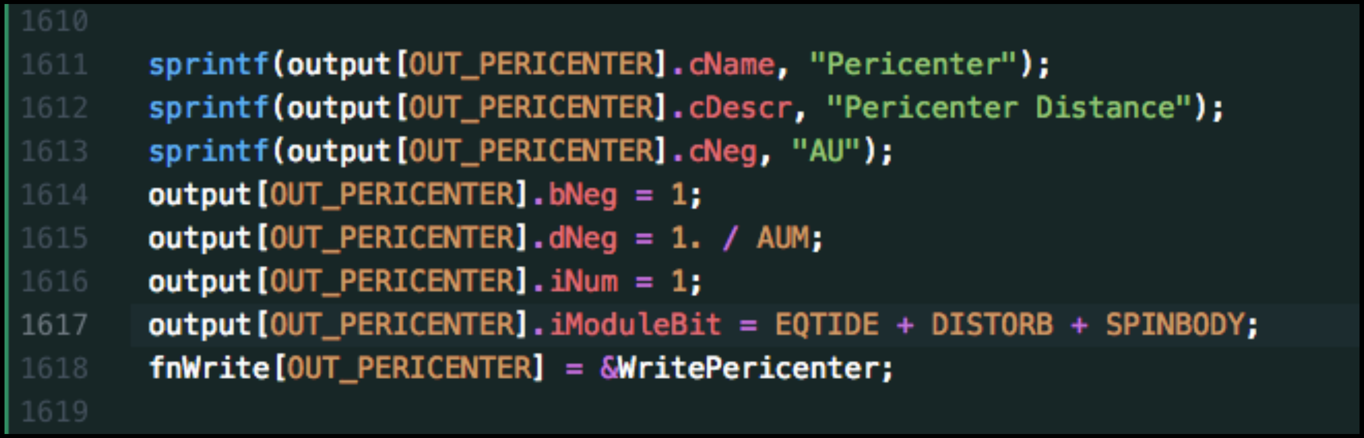
Each output has a block of text that defines its properties

For our new output, we want to follow this format



```

/*
 * End individual write functions
 */

void InitializeOutputGeneral(OUTPUT *output, fnWriteOutput fnWrite[]) {
  /*
   * Age
   */

  sprintf(output[OUT_AGE].cName, "Age");
  sprintf(output[OUT_AGE].cDescr, "System Age");
  sprintf(output[OUT_AGE].cNeg, "Gyr");
  output[OUT_AGE].bNeg       = 1;
  output[OUT_AGE].dNeg       = 1. / (YEARSEC * 1e9);
  output[OUT_AGE].iNum       = 1;
  output[OUT_AGE].iModuleBit = 1;
  fnWrite[OUT_AGE]           = &WriteAge;

```

Step 3: Update InitializeOutput



```
sprintf(output[OUT_PERICENTER].cName, "Pericenter");
sprintf(output[OUT_PERICENTER].cDescr, "Pericenter Distance");
sprintf(output[OUT_PERICENTER].cNeg, "AU");
output[OUT_PERICENTER].bNeg = 1;
output[OUT_PERICENTER].dNeg = 1. / AUM;
output[OUT_PERICENTER].iNum = 1;
output[OUT_PERICENTER].iModuleBit = EQTIDE + DISTORB + SPINBODY;
fnWrite[OUT_PERICENTER] = &WritePericenter;
```

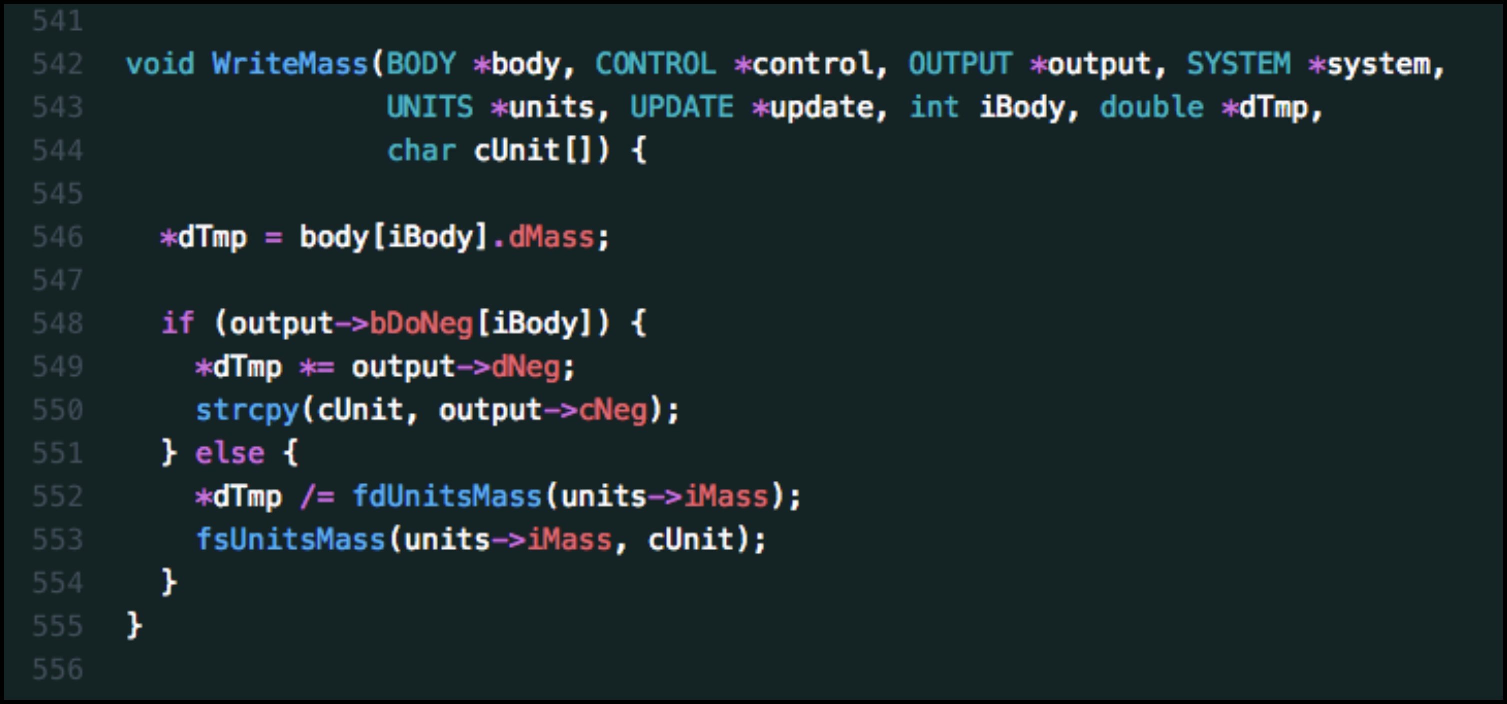
1) Replace the index with the new OUT macro
2) Replace name and description
3) Allow a “negative unit”, set to [au]
4) Only one number is output to the files
5) The output will work for EqTide, DistOrb and SpiNBody
6) The function WritePericenter calculates the value

I think the short description is sufficient, so I didn’t include a long version

Step 4: Write the Write Function

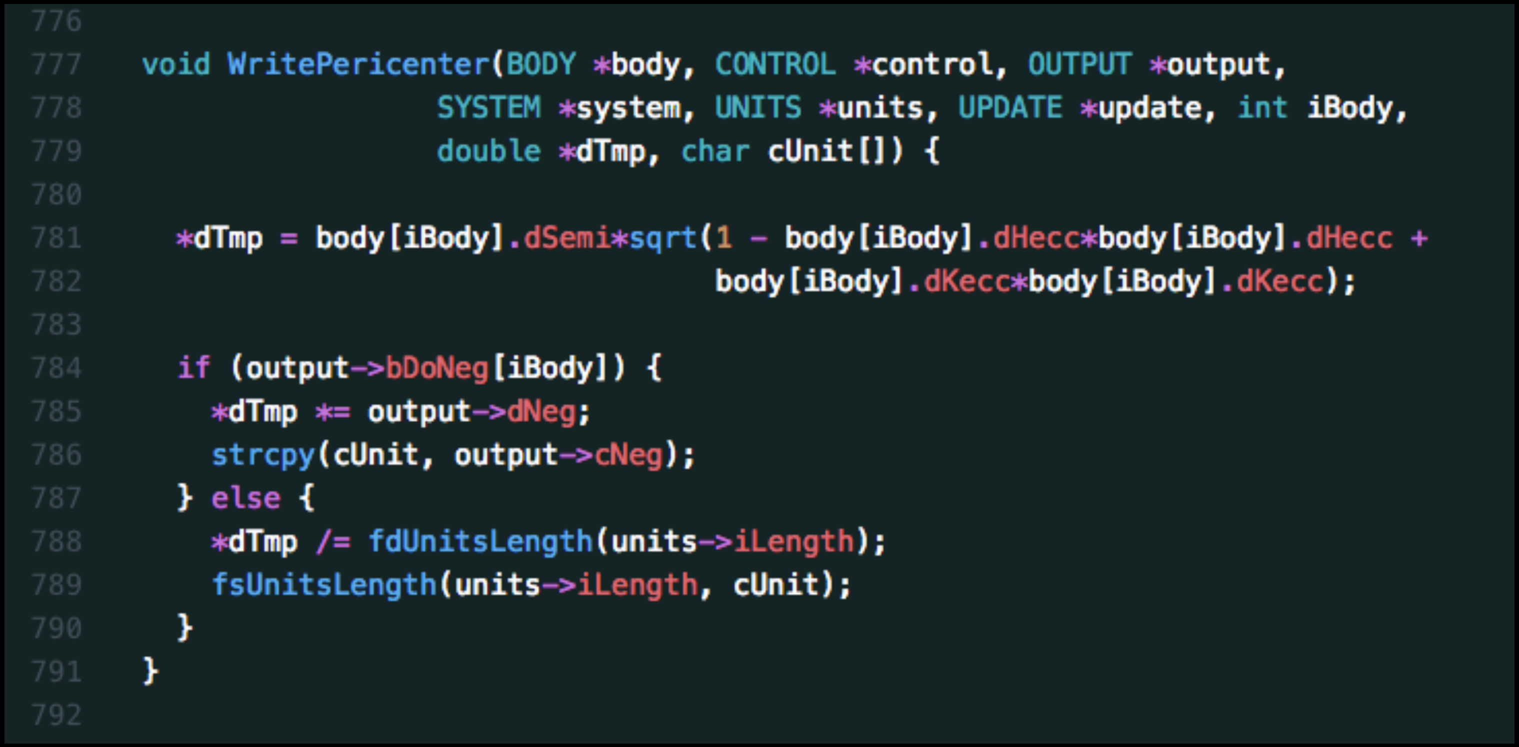
Write functions are part of the fnWrite function pointer array

The argument list must match the typedef, even if we don't need all the data



```
void WriteMass(BODY *body, CONTROL *control, OUTPUT *output, SYSTEM *system,
               UNITS *units, UPDATE *update, int iBody, double *dTmp,
               char cUnit[]) {

  *dTmp = body[iBody].dMass;

  if (output->bDoNeg[iBody]) {
    *dTmp *= output->dNeg;
    strcpy(cUnit, output->cNeg);
  } else {
    *dTmp /= fdUnitsMass(units->iMass);
    fsUnitsMass(units->iMass, cUnit);
  }
}
```

Step 4: Write the Write Function



```
void WritePericenter(BODY *body, CONTROL *control, OUTPUT *output,
                     SYSTEM *system, UNITS *units, UPDATE *update, int iBody,
                     double *dTmp, char cUnit[]) {

  *dTmp = body[iBody].dSemi*sqrt(1 - body[iBody].dHecc*body[iBody].dHecc +
                                 body[iBody].dKecc*body[iBody].dKecc);

  if (output->bDoNeg[iBody]) {
    *dTmp *= output->dNeg;
    strcpy(cUnit, output->cNeg);
  } else {
    *dTmp /= fdUnitsLength(units->iLength);
    fsUnitsLength(units->iLength, cUnit);
  }
}
```

This code will work for DistOrb and EqTide, but not for SpiNBody

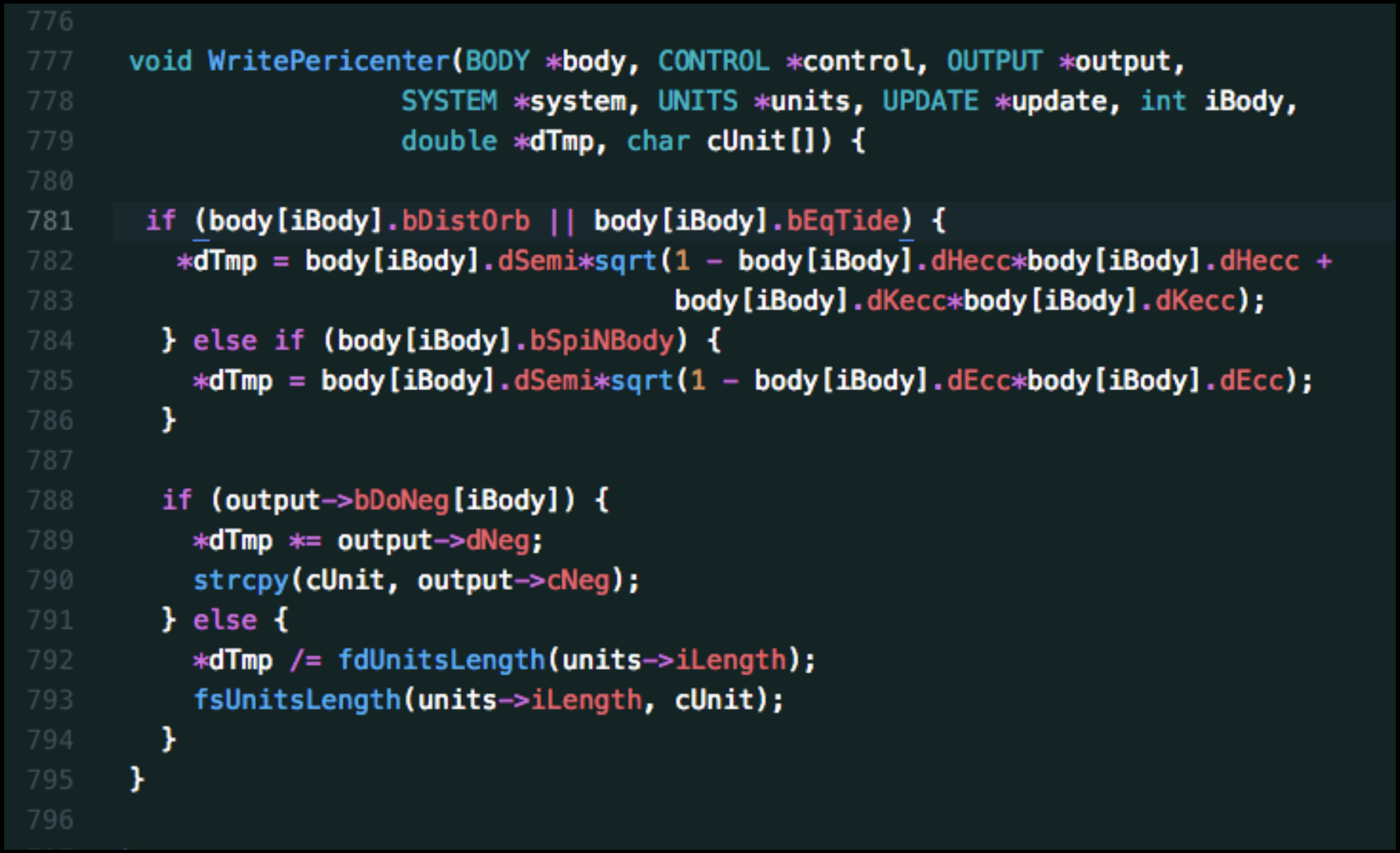
Primary Variables for it are x, y, z, vx, vy, vz

At start of WriteOutput step, code will calculate orbital elements

- but not Hecc and Kecc!

Need to differentiate between DistOrb and SpiNBody!

Step 4: Write the Write Function

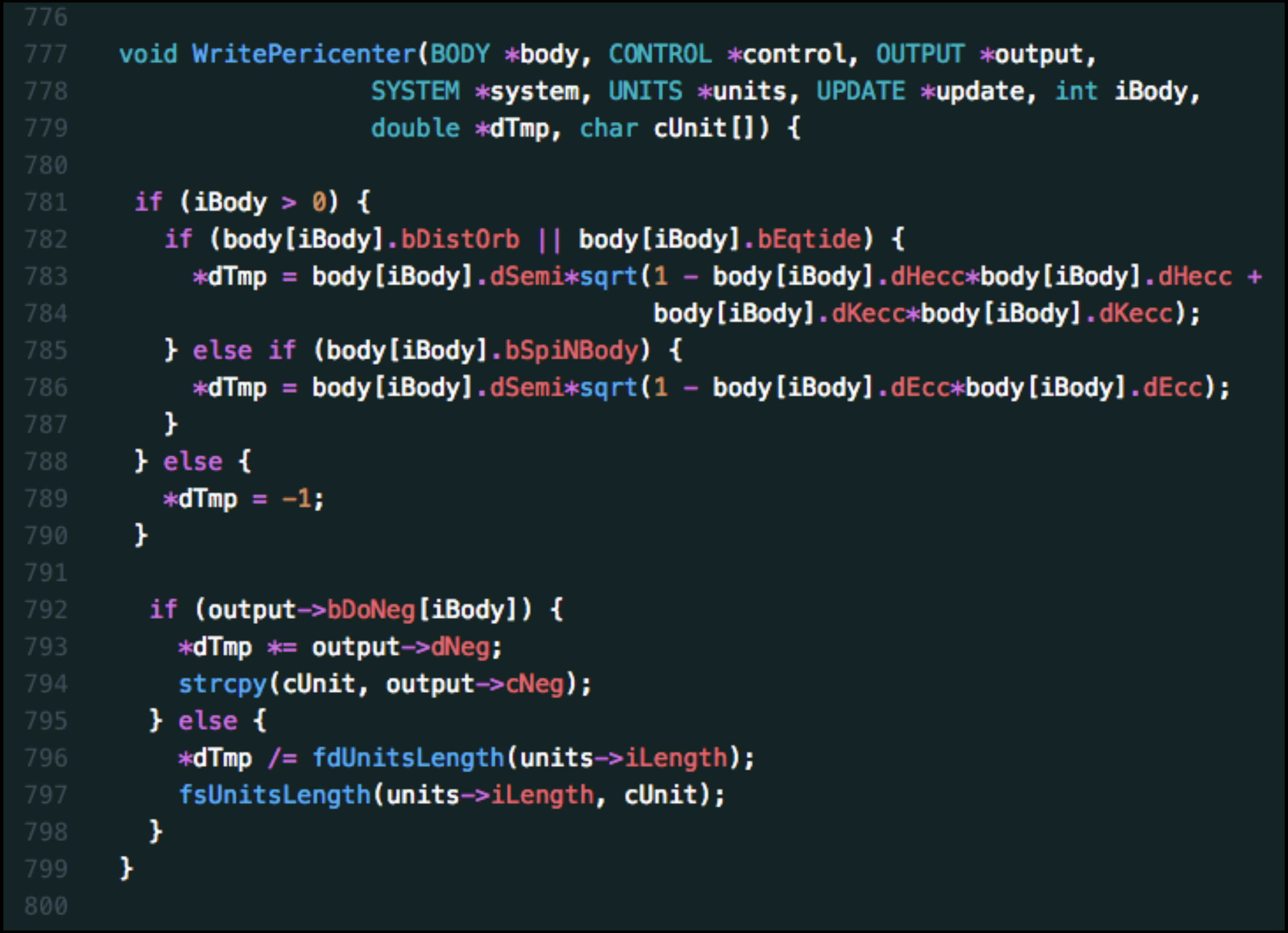


```c
void WritePericenter(BODY *body, CONTROL *control, OUTPUT *output,
                     SYSTEM *system, UNITS *units, UPDATE *update, int iBody,
                     double *dTmp, char cUnit[]) {

  if (body[iBody].bDistOrb || body[iBody].bEqTide) {
    *dTmp = body[iBody].dSemi*sqrt(1 - body[iBody].dHecc*body[iBody].dHecc +
                                   body[iBody].dKecc*body[iBody].dKecc);
  } else if (body[iBody].bSpiNBody) {
    *dTmp = body[iBody].dSemi*sqrt(1 - body[iBody].dEcc*body[iBody].dEcc);
  }

  if (output->bDoNeg[iBody]) {
    *dTmp *= output->dNeg;
    strcpy(cUnit, output->cNeg);
  } else {
    *dTmp /= fdUnitsLength(units->iLength);
    fsUnitsLength(units->iLength, cUnit);
  }
}
```

Now it works for all three modules!

But… the central body is always at (0,0,0) and has no orbital info

Step 4: Write the Write Function



```
void WritePericenter(BODY *body, CONTROL *control, OUTPUT *output,
                     SYSTEM *system, UNITS *units, UPDATE *update, int iBody,
                     double *dTmp, char cUnit[]) {

 if (iBody > 0) {
   if (body[iBody].bDistOrb || body[iBody].bEqtide) {
     *dTmp = body[iBody].dSemi*sqrt(1 - body[iBody].dHecc*body[iBody].dHecc +
                                   body[iBody].dKecc*body[iBody].dKecc);
   } else if (body[iBody].bSpiNBody) {
     *dTmp = body[iBody].dSemi*sqrt(1 - body[iBody].dEcc*body[iBody].dEcc);
   }
 } else {
   *dTmp = -1;
 }

  if (output->bDoNeg[iBody]) {
    *dTmp *= output->dNeg;
    strcpy(cUnit, output->cNeg);
  } else {
    *dTmp /= fdUnitsLength(units->iLength);
    fsUnitsLength(units->iLength, cUnit);
  }
}
```

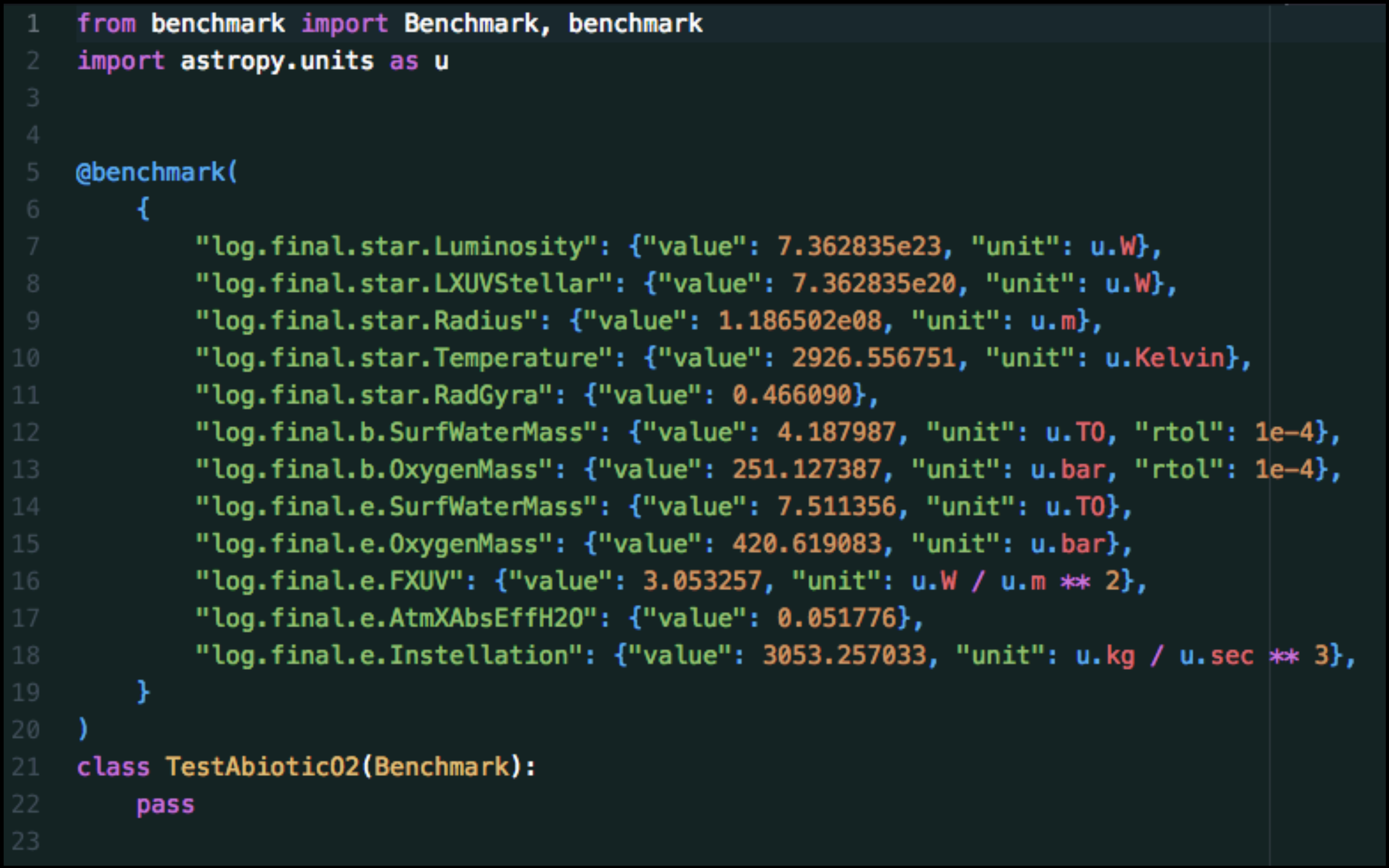
For undefined outputs, set them to -1

And that’s it! The function pointers automatically add this new output!

Adding New Tests

Unit tests are located in the tests/ directory
They are grouped together for convenience

tests/test_AbioticO2.py



```python
from benchmark import Benchmark, benchmark
import astropy.units as u


@benchmark(
    {
        "log.final.star.Luminosity": {"value": 7.362835e23, "unit": u.W},
        "log.final.star.LXUVStellar": {"value": 7.362835e20, "unit": u.W},
        "log.final.star.Radius": {"value": 1.186502e08, "unit": u.m},
        "log.final.star.Temperature": {"value": 2926.556751, "unit": u.Kelvin},
        "log.final.star.RadGyra": {"value": 0.466090},
        "log.final.b.SurfWaterMass": {"value": 4.187987, "unit": u.TO, "rtol": 1e-4},
        "log.final.b.OxygenMass": {"value": 251.127387, "unit": u.bar, "rtol": 1e-4},
        "log.final.e.SurfWaterMass": {"value": 7.511356, "unit": u.TO},
        "log.final.e.OxygenMass": {"value": 420.619083, "unit": u.bar},
        "log.final.e.FXUV": {"value": 3.053257, "unit": u.W / u.m ** 2},
        "log.final.e.AtmXAbsEffH2O": {"value": 0.051776},
        "log.final.e.Instellation": {"value": 3053.257033, "unit": u.kg / u.sec ** 3},
    }
)
class TestAbioticO2(Benchmark):
    pass

```

To add a new test, simply add a new line to the code
To perform the test: pytest test_AbioticO2.py

Adding New Tests

Pericenter applies to about a dozen tests, but we need only add unit tests to span the functionality

We need at least 1 tests for EqTide, DistOrb and SpiNBody

EqTide has two models: CPL and CTL

DistOrb has two models: 2nd and 4th order ("LL2" and "RD4")
- (But no tests yet for LL2)

SpiNBody has only 1 model

=> 4 single-module tests

Adding New Tests

DistOrb and EqTide can couple, so we need additional tests
- RD4 + CPL (no existing tests for RD4 + CTL yet, either)

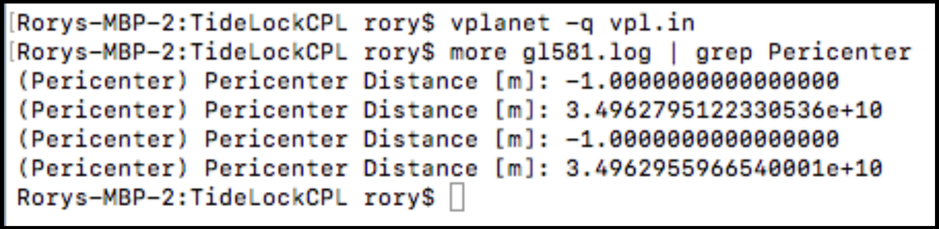
Currently SpiNBody and EqTide have not been coupled

So we can easily add 4 single module tests + 1 multi-module tests

We can run the tests with our new code, and copy/pastecd the results to the Python script

Let's add Pericenter to the following tests:
- TideLockCPL
- TideLockCTL
- SS_SpiNBody
- SSDistOrbDistRot
- ApseLock (RD4 + CPL)

Adding New Tests



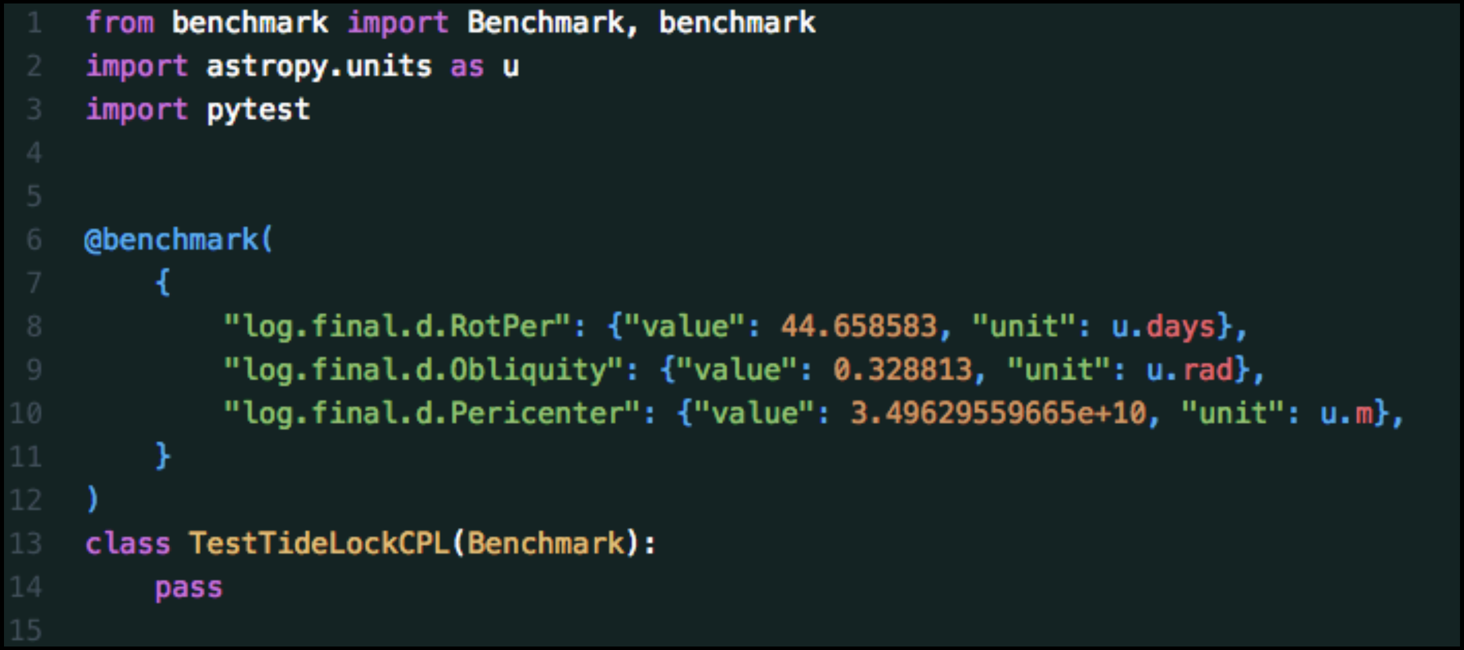
```
[Rorys-MBP-2:TideLockCPL rory$ vplanet -q vpl.in
[Rorys-MBP-2:TideLockCPL rory$ more gl581.log | grep Pericenter
(Pericenter) Pericenter Distance [m]: -1.000000000000000
(Pericenter) Pericenter Distance [m]: 3.4962795122330536e+10
(Pericenter) Pericenter Distance [m]: -1.000000000000000
(Pericenter) Pericenter Distance [m]: 3.4962955966540001e+10
Rorys-MBP-2:TideLockCPL rory$ 
```

} Initial values

} Final values

The final pericenter distance for the planet (d) is 3.4926…e+10

We then add this to the unit test:



```python
from benchmark import Benchmark, benchmark
import astropy.units as u
import pytest


@benchmark(
    {
        "log.final.d.RotPer": {"value": 44.658583, "unit": u.days},
        "log.final.d.Obliquity": {"value": 0.328813, "unit": u.rad},
        "log.final.d.Pericenter": {"value": 3.49629559665e+10, "unit": u.m},
    }
)
class TestTideLockCPL(Benchmark):
    pass

```

It's not necessary to use all 16 digits of precision

We then follow this procedure for the other 4 tests

Issuing the Pull Request

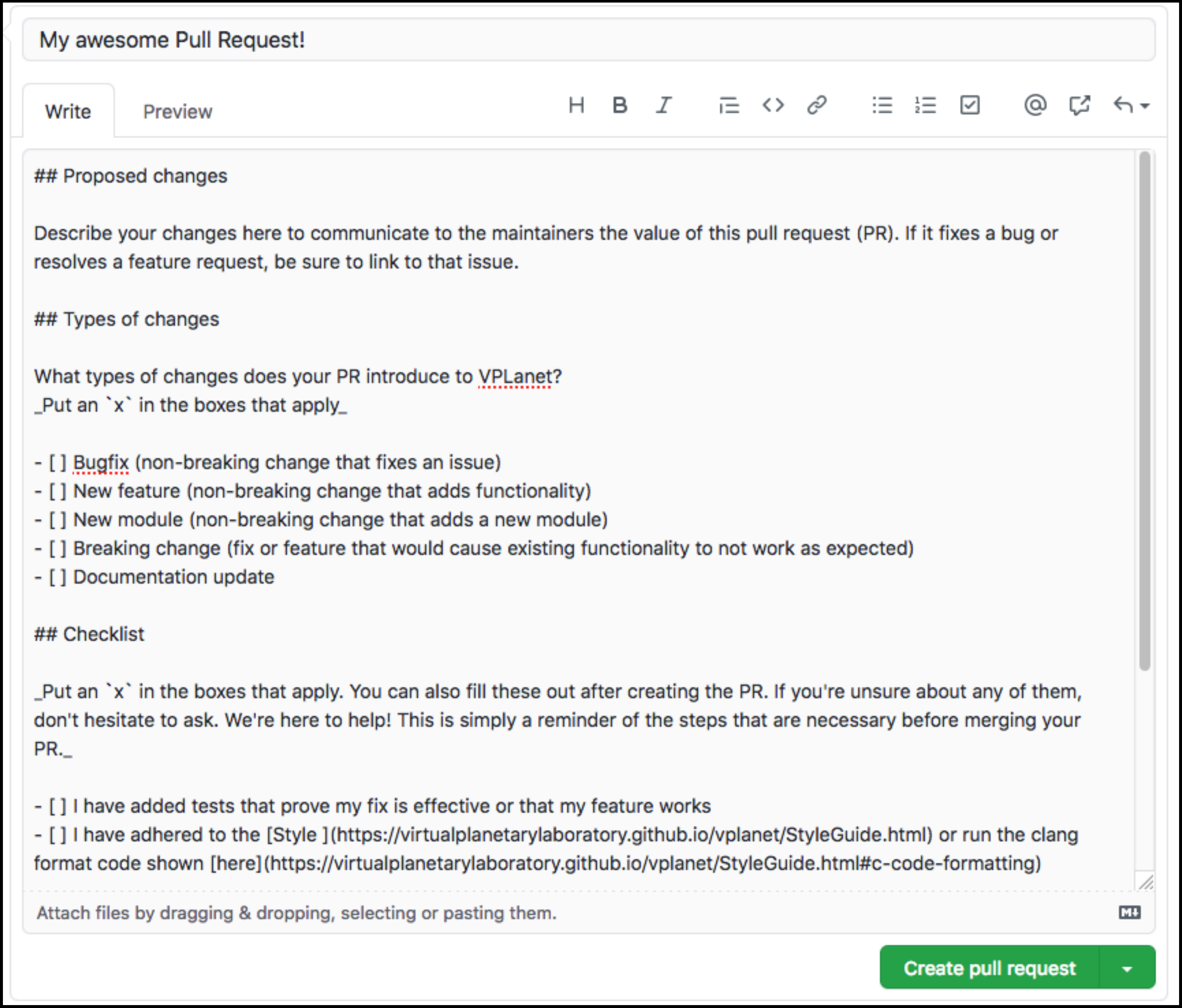
Unit tests are the most critical part of the PR as they maintain code integrity

But there are additional steps:

- Adhering to the style guide (clang-format automates this)
- Running Valgrind and address-sanitizer and fix memory leaks
- Update documentation (if necessary)
    - Note that help and online documentation will automatically update with your new output
- Adding/updating examples (if necessary)

Once these steps are done, you are ready to issue your PR!

Issuing the Pull Request



My awesome Pull Request!

```
## Proposed changes

Describe your changes here to communicate to the maintainers the value of this pull request (PR). If it fixes a bug or
resolves a feature request, be sure to link to that issue.

## Types of changes

What types of changes does your PR introduce to VPLanet?
_Put an `x` in the boxes that apply_

- [ ] Bugfix (non-breaking change that fixes an issue)
- [ ] New feature (non-breaking change that adds functionality)
- [ ] New module (non-breaking change that adds a new module)
- [ ] Breaking change (fix or feature that would cause existing functionality to not work as expected)
- [ ] Documentation update

## Checklist

_Put an `x` in the boxes that apply. You can also fill these out after creating the PR. If you're unsure about any of them,
don't hesitate to ask. We're here to help! This is simply a reminder of the steps that are necessary before merging your
PR._

- [ ] I have added tests that prove my fix is effective or that my feature works
- [ ] I have adhered to the [Style ](https://virtualplanetarylaboratory.github.io/vplanet/StyleGuide.html) or run the clang
format code shown [here](https://virtualplanetarylaboratory.github.io/vplanet/StyleGuide.html#c-code-formatting)
```

Attach files by dragging & dropping, selecting or pasting them.

Homework

Try adding an output yourself!

You're here because you want to bend VPLanet to your will, so give it a try while it's fresh in your mind

If you have issues/questions, e-mail me (rkb9@uw.edu), and I'll try to answer them

You can also ask on Friday.

Good luck!