COP-3402 Systems Software
10.03 Thu

Recursion

Pseudocode for Factorial:
If we have factorial(5);
Consider factorial(5) as a function. Somehow there'll be a recursive call to factorial(4) times 5, a call to factorial(3) times 4 times 5, and so on…

You always have the base case and recursive step.
By doing it over an example, it gives us a hint at what recursive step looks like.

The scheme

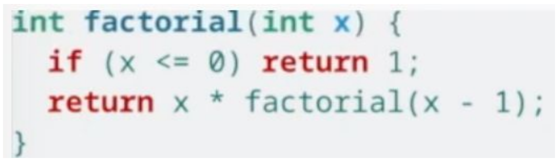
```
factorial(x):
        //recursive_step
                factorial
        //base case
```

Assume *factorial* is already written inside the function, so we can just say *return factorial.*

//requires x>0, x int

```
factorial(int x):
        if (x == 1 or x == 0 ) return 1   //base case
        return x * factorial(x-1)         //recursive step
```

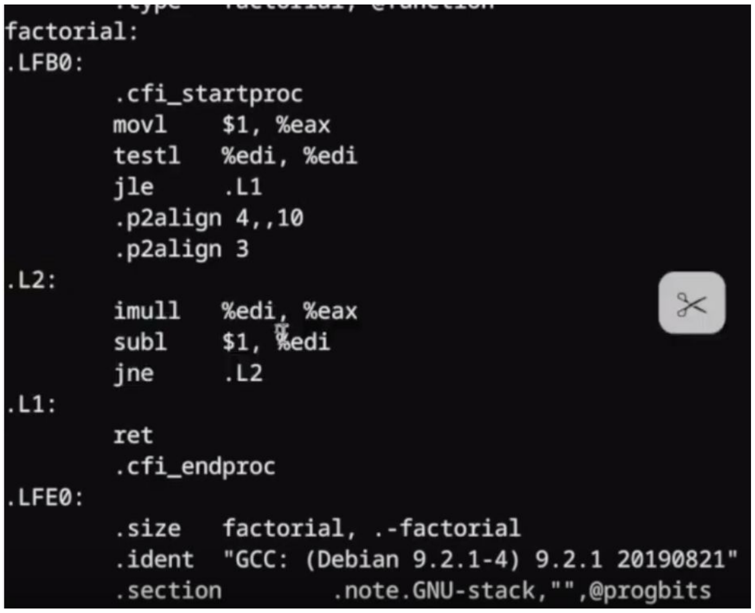
Computational Complexity: O(n)



```
int factorial(int x) {
  if (x <= 0) return 1;
  return x * factorial(x - 1);
}
```

Command window:

```
emacs fact.s
gcc -S -00 fact.s
less fact.s
```



```
factorial:
.LFB0:
        .cfi_startproc
        movl    $1, %eax
        testl   %edi, %edi
        jle     .L1
        .p2align 4,,10
        .p2align 3
.L2:
        imull   %edi, %eax
        subl    $1, %edi
        jne     .L2
.L1:
        ret
        .cfi_endproc
.LFE0:
        .size   factorial, .-factorial
        .ident  "GCC: (Debian 9.2.1-4) 9.2.1 20190821"
        .section        .note.GNU-stack,"",@progbits
```

REVIEW: For command window guide, refer to the cop3402fall19/syllabus/projects

For non-vagrant users:

```
sudo apt-get update
sudo apt-get install clang llvm git make python3 python3-distutils
```

Then,
Configuring git:
Setup your name and email (use the email associated with your GitHub account).

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

Set your editor for commit messages

```
sudo apt-get install nano
git config --global core.editor "nano"
```

Assume you cloned syllabus repository already.

```
git clone https://github.com/cop3402fall19/syllabus.git
```

If you have vagrant setup, go to the mapped `/vagrant` directory (this is mapped to your host machine's syllabus/projects directory)

```
cd /vagrant # for vagrant users
cd ~/ # otherwise
```

Then under the syllabus/projects,

```
git clone https://github.com/cop3402fall19/project-USERID.git
```

Getting the grading scripts

Go to your home directory

```
cd
```

Clone the repository

```
git clone https://github.com/cop3402fall19/grader-scripts.git
```

Go to your local clone and pull

```
cd ~/syllabus
git pull
cd ~/grader-scripts
git pull
```

File system structure

If you are using vagrant, you should have these directories:

```
/vagrant/project-USERID
/home/vagrant/syllabus
/home/vagrant/grader-scripts
```

Otherwise you should have these:

```
/home/USER/project-USERID
/home/USER/syllabus
/home/USER/grader-scripts
```

Setting up your project Makefile

```
cd /vagrant/project-USERID # for vagrant users
cd ~/project-USERID # otherwise
```

Copy the provided Makefile into your source code repository.

```
cp ~/syllabus/projects/make/Makefile ./
```

Commit the Makefile to your repository. A commit saves changes to the source repository in a log.

```
git add Makefile
git commit Makefile
```

`git commit` will open an editor. Enter a message at the top describing the change. Exit using `ctrl-x`, then hit `y` to confirm saving, and finally hit the `enter` key to confirm the file name.

Create your C source code

```
touch simplec.c
```

As you code your program in this file, Then add it to the repository to make the changes

```
git add simplec.c
git commit simple.c
git status

git push
```

Running your compiler

Every time you make changes to your source code, run `make` to recompile it.

```
make
```

Run your project like this

```
./simplec ~/syllabus/projects/tests/proj0/all.simplec
```

This should take the `all.simplec` test file and print the resulting LLVM IR to your terminal window.

Convert the LLVM IR to machine code:

```
./simplec ~/syllabus/projects/tests/proj0/all.simplec > /tmp/all.ll
```

The `> /tmp/all.ll` means the output will be written, i.e., redirected, to the `all.ll` file in the `/tmp` directory and will not show up in your terminal window. You can quickly view the contents of this file with `cat`

```
cat /tmp/all.ll
```

```
clang -o /tmp/all /tmp/all.ll  # convert to machine code
```

```
/tmp/all # run the program
```

```
/tmp/all > /tmp/all.out  # save the output to all.out
```

```
diff ~/syllabus/projects/tests/proj0/all.groundtruth /tmp/all.out
```

The complete set of instructions to compile and run a SimpleC program using your compiler is this

```
./simplec ~/syllabus/projects/tests/proj0/all.simplec > /tmp/all.ll
```

```
clang -o /tmp/all /tmp/all.ll
```

```
/tmp/all > /tmp/all.out
```

```
diff ~/syllabus/projects/tests/proj0/all.groundtruth /tmp/all.out
```

If the final `diff` showed no differences or no errors, that means your compiler worked correctly on that specific test case.

Run the Grader Script for all test cases

```
python3 ~/grader-scripts/testcasesScript.py ./ ~/syllabus/projects/tests/proj0/
```

```
# the following are all the commands run by this test script.  you can cut-and-paste
them to run them by hand.
```

```
# building your simplec compiler
```

```
make
```

```
# TESTING ../syllabus/projects/tests/proj0/all.simplec
```

```
/home/paul/research/teaching/cop3402fall19/grader-scripts/compile.sh ./simplec
../syllabus/projects/tests/proj0/all.simplec
```

```
# PASSED

/home/paul/research/teaching/cop3402fall19/grader-scripts/run.sh
../syllabus/projects/tests/proj0/all.ll

# PASSED

# TESTING ../syllabus/projects/tests/proj0/sub.simplec

/home/paul/research/teaching/cop3402fall19/grader-scripts/compile.sh ./simplec
../syllabus/projects/tests/proj0/sub.simplec

# PASSED

/home/paul/research/teaching/cop3402fall19/grader-scripts/run.sh
../syllabus/projects/tests/proj0/sub.ll

# ERROR run.sh failed on ../syllabus/projects/tests/proj0/sub.ll
```

This will use your `simplec` program to compile a SimpleC program to LLVM IR.

```
~/grader-scripts/compile.sh project-USERID/simplec
~/syllabus/projects/tests/proj0/all.simplec
```

The output will be in `all.ll` in the same path as the `all.simplec`.

```
~/grader-scripts/run.sh ~/syllabus/projects/tests/proj0/all.ll
```

The output will be in `all.out` in the same path as the `all.simplec`. `run.sh` will automatically compare `all.out` to `all.groundtruth` if available.

Submitting your project

```
git commit simplec.c

git push

git tag proj0

git push --tags
```

Verify your submission:

```
https://github.com/cop3402fall19/project-USERID/releases
```

Resubmitting your project

```
git tag -f proj0

git push -f --tags
```

Sanity Check:

```
git clone https://github.com/cop3402fall19/project-USERID.git /tmp/test-project

cd /tmp/test-project

git checkout proj0  # use the appropriate tag for the project

make

# run tests

cd

rm -rf /tmp/test-project # this will destroy the temporary copy, so do not put
anything important in here
```