CIS*1500 LAB #1

TA: Alliyya Mo

Lab Goals

- Changing SOCS password
- Navigating the command line
- Basic Unix Commands
- Using Nano as a text editor
- Raspberry Pi Set-up in rm 2418

Changing SOCS Password

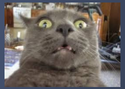
Your current username is the first part of your uoguelph email address

Your current password is your 7-digit student number(including the zero, ex 0123123)

```
User:~$ passwd
```

Basic Unix Commands

- ls
- mkdir
- touch
- rm
- cd

- clear
- mv
- cp
- cat 

- man (Your future BFF)

Setting up Raspberry Pi

Plug in power cord last!

Steps:

1. Plug in peripherals first (mouse, keyboard, HDMI)
2. Plug in ethernet cable
3. Finally power cord!
4. :)

Nano navigation

● create a file		
● open file	nano -c test.txt	
● save file	ctrl-o	*(Not ctrl-s)*
● exit	ctrl-x	

Things To Remember for Next Lab

- Regular labs will be in 2418!
- Always bring your Pi to your labs!
- ...Seriously, don’t forget your Pi’s!

ALSO: Do you have anything due tomorrow? By 9 am?

CIS*1500 Lab 2

Introduction to Programming

TA: Alliyya Mo

Today's Lab

- Review Linux commands
- Pi Setup
- Create your first Hello World program
- Compile gcc with and without flags(-std=c99 -Wall -pedantic)
- Run your first program
- Explore printf(w. variables if time permits)
- Also how many did textbook readings!

Raspberry Pi Setup

1. Put the SD card into the SD card slot on the Raspberry Pi (fits one way)
2. Plug in keyboard and mouse into PI
3. Turn on the monitor and connect your HDMI cable from your Pi to your monitor
4. Plug in the Ethernet cable into the Ethernet
5. Finally plug in the micro USB power supply. This will turn on and boot your Raspberry Pi.

NOTE: Make sure you have NOOBS preinstalled on the SD card otherwise come see me!

Quick Review

● ls	-	list files in a directory
● cd	-	change directories
● pwd	-	print working directory(where am I?)
● cp	-	copy files(dest/src?)
● mv	-	move(rename) files
● rm	-	remove files/directories(rm -r)
● touch	-	new file
● man	-	manual pages

Touched upon in Lecture not in Lab + Extra!

- history (!)
- grep
- mv (to rename files)
- more

Try man -k directory

Warm-up Exercise

1. Change directories into your ricky folder (if it doesn't exist, create it)
2. Create a folder, name it joe
3. Create another folder named mo
4. Move joe folder to mo
5. Go into joe's directory
6. create a new file named helloWorld.c
7. Open helloWorld.c with nano

1. cd cis1500
2. mkdir labs
3. mkdir lab2
4. mv lab2 labs
5. cd labs/lab2
   or
   cd labs
   cd lab2
6. touch helloWorld.c
7. nano helloWorld.c

```c
#include <stdio.h>
int main()
{
    printf("hello world!\n");
    return 0;
}
```

#include <stdio.h>

- How we include libraries into your program
- Libraries contain functions and instructions for common tasks like printing or sqrts
- This one is called standard I/O (input/output)
- It contains printing, file reading and keyboard input functions that are vital to most programs in C

int main(void) { return(0);}

The program starts by executing a function called main. A function is a list of statements

"{" and "}" are called braces, denoting a list of statements. main's statements appear between braces.

A statement is a program instruction. Each statement usually appears on its own line.

return(0);

Each program statement ends with a semicolon ";"

The main function and hence the program ends when the return statement executes. The 0 in return 0; tells the operating system that the program is ending without an error.

printf(“Hello World\n”);

PrintFormatted function is simply going to print out text , “Hello World”, to our screen

The \n means a newline will be printed after Hello World

\t means a tab will be printed

Commenting!

- As you write more complex code you’re going to want to document your code to avoid future confusion
- Code tells you how, comments will tell you why
- // is for a single line comment(everything after till the end of that line will be ignored)
- /**/ is for multiline comments (everything between /* and */ will be ignored by the

Our Source code(helloWorld.c)

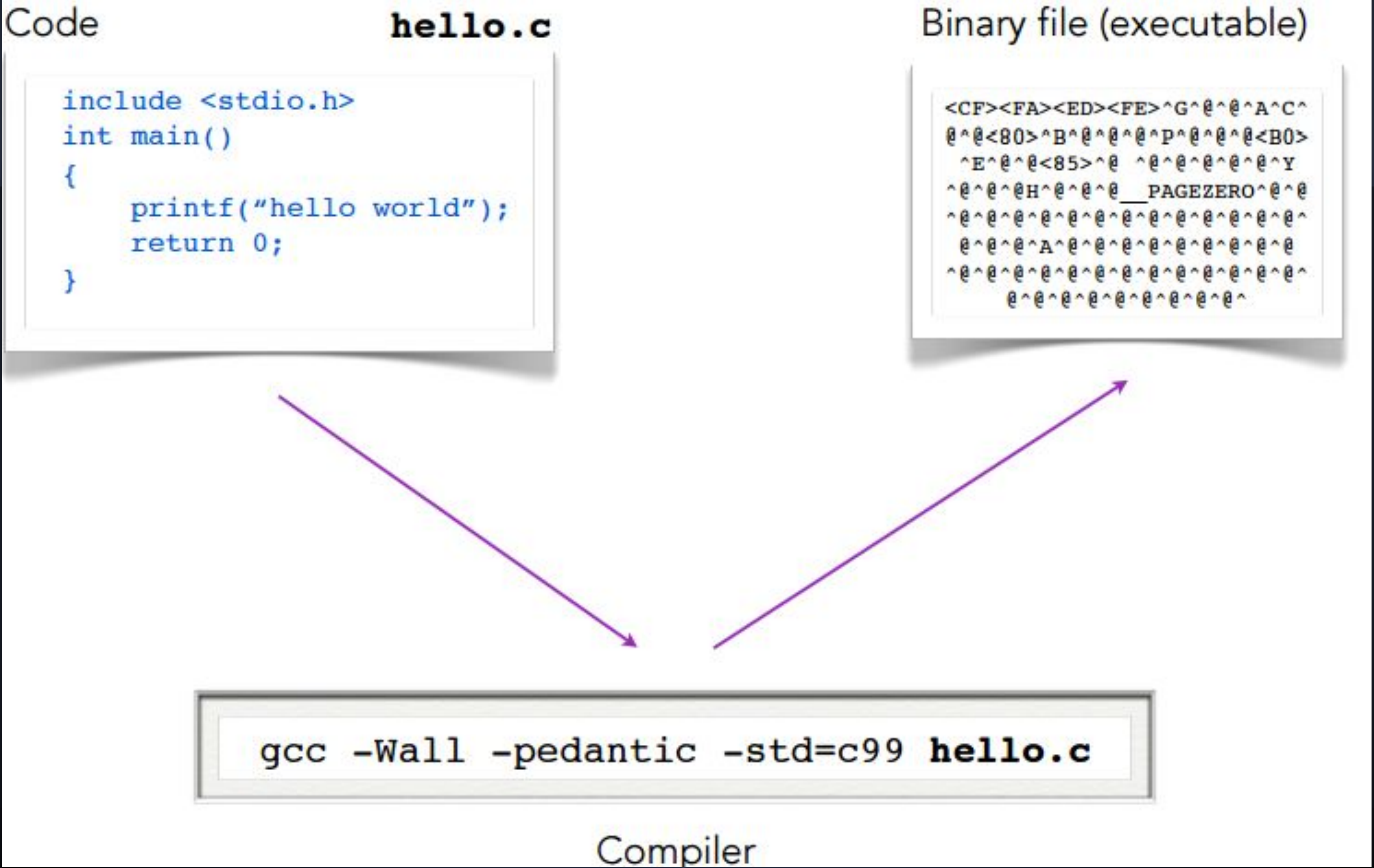
```
/*This is our hello world program*/
#include <stdio.h>
int main(void)
{
    printf("Hello World!!\n"); // will print hello world
    return 0;
}
```

Compiling your first program! (basic)

```
gcc <files> <options>
```

```
gcc helloWorld.c
```

- this will compile it and create an executable named a.out (type ls and cat a.out)



```
include <stdio.h>
int main()
{
    printf("hello world");
    return 0;
}
```

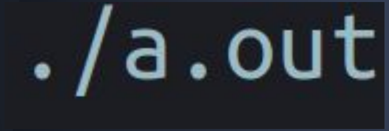
```
<CF><FA><ED><FE>^G^@^@^A^C^
@^@<80>^B^@^@^@^P^@^@^@<B0>
^E^@^@<85>^@ ^@^@^@^@^@^Y
^@^@^@H^@^@^@__PAGEZERO^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^A^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^
```

```
gcc -Wall -pedantic -std=c99 hello.c
```

Running your first program

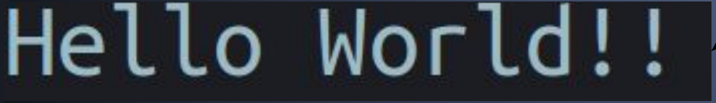
Just type:

./a.out



```
./a.out
```

Our Program running!
This is its output!



```
Hello World!!
```

Remember hitting tab to autocomplete is your friend!

CIS*1500 standards for compilation

```
gcc <files> <options>
gcc helloWorld.c -std=c99 -Wall -pedantic -o hello
```

-std=c99: C99 standards that we will be using in C

-Wall: List all the warning generated during compilation

-pedantic: more warnings!(From being more pedantic)

-o: renames our executable something else

Lab Exercise

1. If you haven’t done already, type your Hello World program! (Don’t forget anything!)
2. Compile with just gcc
3. cat ./a.out (That’s our executable!)
4. Run with ./a.out
5. Compile with flags this time, and rename your executable Hello(-o hello)
6. Run with ./hello

Try playing around with code!

- Make it print out whatever you want
- use /t and /n in your printf
- do multiple printfs
- try compiling missing a semicolon, what happens?
- add comments (multi and single)

Bonus Exercise

Play around with unix commands and man pages

Look up 3 different commands and try an assortment of the options listed in the man pages

ex. `ls -l -a` also can be written as `ls -la`

More Exercises

Go onto bucky and go through Intro to C Syntax lesson

If time permits we'll talk about Variables and play more with printf

Need Extra help?

- Free tutoring offered by TAs! (Book an appointment on bucky) (also see me after Lab)
- Drop-in help hours (right after your Lecture!) (11:30,2:30 and 5:30) (Tuesday & Thursday)

All meetings will take place in room 001/002 in the basement of Reynolds.

Got a question?

- Ask me! :)
- Post on the Forums: forum.socs.uoguelph.ca
- Email us: cis1500@socs.uoguelph.ca

Protip: Search the forums and bucky before making a post or sending an email!

Or are you Incredibly Bored?

And looking for a Challenge? Or something new?

Come see me after lab!

Sites To Check To Stay up to Date

Course Website: bucky.socs.uoguelph.ca

SOCS Forums: forum.socs.uoguelph.ca

Textbook: zybooks.com

Reminders!

- **Complete assigned textbook readings** before 9 am Tomorrow(Tuesday)!
- Complete academic integrity quiz on Moodle(see bucky for instructions!)
    - must be completed by October 10th
- **Protip: Finish assigned readings before your Lab!** (The lab will feel like a light breeze if you do that)

CIS*1500 Lab 3

Introduction to Programming
TA: Alliyya Mo

Today’s Lab

- Review gcc with and without flags(-std=c99 -Wall -pedantic)
- Review variables and identifiers
- Review printf
- Review scanf
- scanf and printf exercises
- Style matters!!
- A bit of debugging practice
- ***Side-Quest #1 for those interested in a fun challenge!***

Raspberry Pi Setup

1. Put the SD card into the SD card slot on the Raspberry Pi (fits one way)
2. Plug in keyboard and mouse into PI
3. Turn on the monitor and connect your HDMI cable from your Pi to your monitor
4. Plug in the Ethernet cable into the Ethernet
5. Finally plug in the micro USB power supply. This will turn on and boot your Raspberry Pi.

NOTE: Make sure you have NOOBS preinstalled on the SD card otherwise come see me!

Warm-up Exercise (5 min)

1. Change directories into your cis1500 folder
2. Change directories into your labs folder
3. Create a directory named lab3
4. Change directories into lab3 folder
5. create a new file named testingWorld.c
6. Open testingWorld.c with nano
7. Recreate your helloWorld program! (Let's see how much you actually remember)
8. Compile your testingWorld.c and run it!

1. cd cis1500
2. cd labs
3. mkdir lab3
4. cd lab3
5. touch testingWorld.c
6. nano testingWorld.c
7. Some witchcraft(coding)
8. gcc testingWorld.c -Wall -std=c99 -pedantic -o test
9. ./test

```c
#include <stdio.h>
int main()
{
    printf("hello world!\n");
    return 0;
}
```

Review helloWorld.c

```
/*This is the rosetta stone of programming,
	It will simply output Hello World to standard output
*/
#include <stdio.h>
int main(void)
{
	printf("Hello World\n"); //prints hello world
	return(0);
}
```

Bonus Exercise for those comfortable with if-statements already!

Create a basic calculator.(+,-,*,/,^)

Allow the user to select an operation and have them enter the required inputs and then display the answer.

Hint: What data type will be more precise?

Any libraries you might need?

Bonus: Do you know loops? Allow the user to return to the menu after receiving their answer

Review Hello world (Quick game of I-Spy)

```
#include <stdio >


    main(void)
{
    print ("Hello world")
return(0);
```

```
#include <stdlib.h>


int main(void)
{
    print ("Hello world")
    return(0)
}
```

The semicolon!

Each program statement ends with a semicolon ";"

How well do you know GCC?

1. `gcc file.c -wall -pedantic -std=c90 -o hello`
2. `gcc file.c -pedantic -wall -std=c99 -o file.c`
3. `gcc file.c -Wall -std=c99 -o hello`
4. `gcc file -std=c99 -Wall -pedantic -o hello.c`
5. `gcc file.c -Wall -Pedantic -Std=c99 -O hello`
6. `gcc file.txt -Wall -pedantic -std=c99 -o hello`
7. `gcc -Wall -pedantic -std=c99 -o hello`

Yeah... It was none, But maybe one of these?

- ~~gcc test.c [illegible]d=c99 -o test.c~~
- gcc test.c -Wall -pedantic -std=c99 -o test

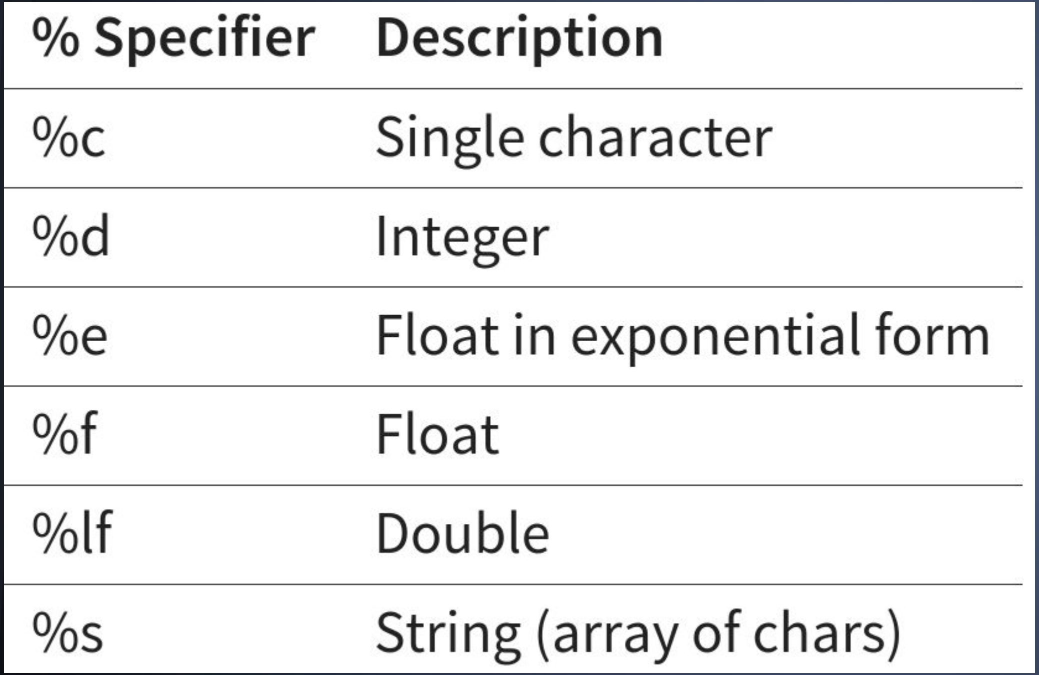
Variables

```
char letter;      //character
int number;       //integer
float decimal;  //floating point number(3.145)
double longerDec;     //longer floating number(more
precise than a float)
```

APPROPRIATE IDENTIFIERS

- Both concise, and descriptive
- Short names are easier to type, but suffer from a lack of meaning (i, a, r)
- Descriptive names are more meaningful (index, array, ratio)
- Long names are too difficult to use and prone to errors when typing.
  - e.g. *boilingpointofwater*
- Two word identifiers can be created by using camelCase to join words together without spaces.
  - e.g. *countOdd, airTemperature, bacteriaGrowth*

printf and Format Specifiers

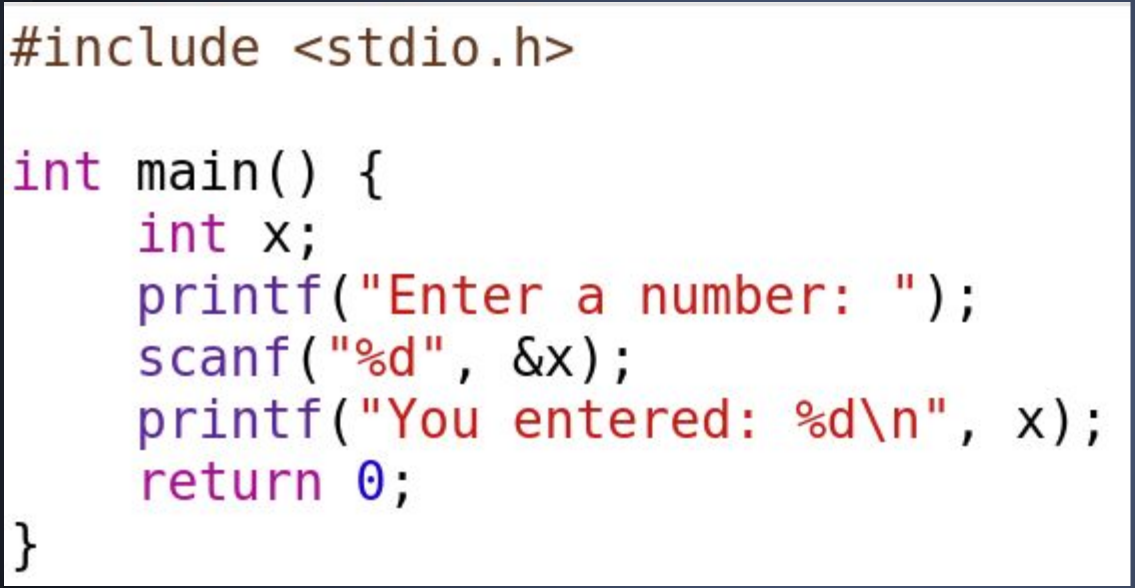


% Specifier	Description
%c	Single character
%d	Integer
%e	Float in exponential form
%f	Float
%lf	Double
%s	String (array of chars)

ex.

```
int num =5;
printf(“%d\n”,num);
```

scanf



```c
#include <stdio.h>

int main() {
    int x;
    printf("Enter a number: ");
    scanf("%d", &x);
    printf("You entered: %d\n", x);
    return 0;
}
```

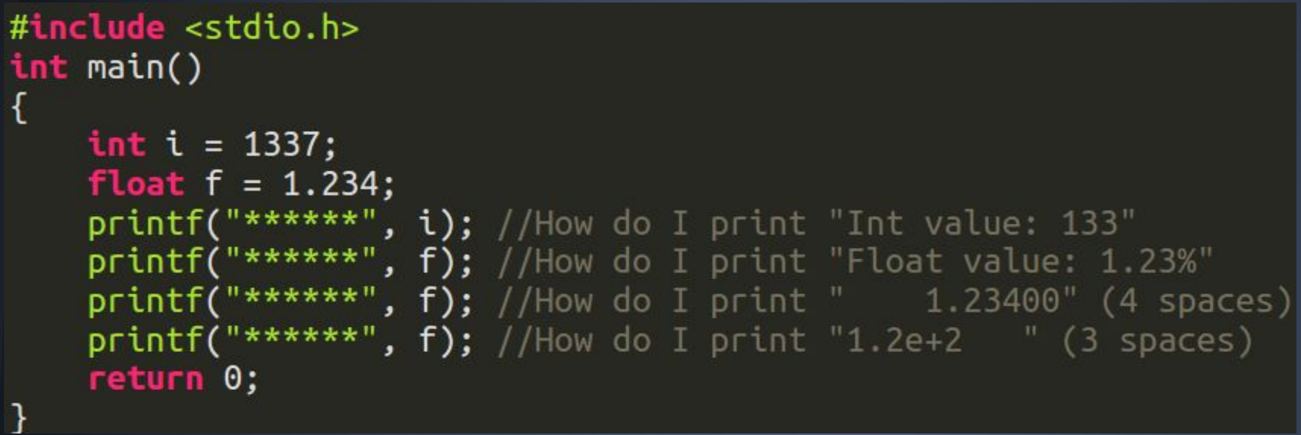
printf and scanf warm-up

- create a new file in your lab3 folder(warmUp.c)
- create a program that asks the user:
  - Favourite letter
  - Their age
  - First 3 digits of pi

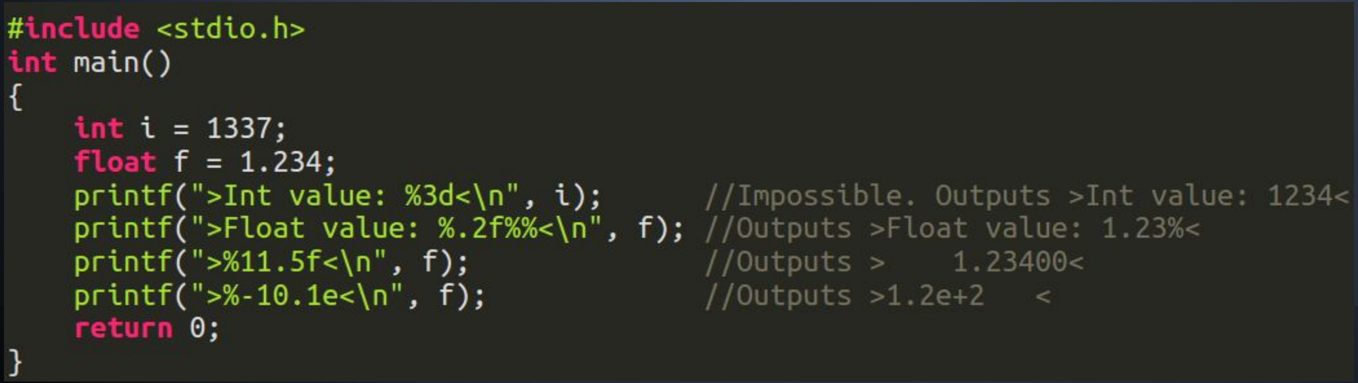
Formatting using printf & field width

Statement	Output(⊔ == spaces)	Info
printf(%d", i);	123	field width 3
printf(%05d", i);	00123	field width 5; padded with 0
printf(%f", x);	32.178658	precision 6 by default
printf(%.3f", x);	32.179	precision 3
printf(%.3e", x);	3.218e+01	same as f, but with e format
printf(%10.3f", x);	⊔⊔⊔⊔32.179	precision 3, field width 10
printf(%-10.3f", x);	32.179⊔⊔⊔⊔	precision 3, left field width 10

Try it yourself!



```c
#include <stdio.h>
int main()
{
    int i = 1337;
    float f = 1.234;
    printf("******", i); //How do I print "Int value: 133"
    printf("******", f); //How do I print "Float value: 1.23%"
    printf("******", f); //How do I print "    1.23400" (4 spaces)
    printf("******", f); //How do I print "1.2e+2   " (3 spaces)
    return 0;
}
```



```c
#include <stdio.h>
int main()
{
    int i = 1337;
    float f = 1.234;
    printf(">Int value: %3d<\n", i);      //Impossible. Outputs >Int value: 1234<
    printf(">Float value: %.2f%%<\n", f); //Outputs >Float value: 1.23%<
    printf(">%11.5f<\n", f);              //Outputs >    1.23400<
    printf(">%-10.1e<\n", f);             //Outputs >1.2e+2   <
    return 0;
}
```

Exercise #1: How fast can you finish this

- create a program that asks the user for the distance travelled and the amount of time taken
- then calculate the velocity and output it to the screen, with only 3 decimal places.
- *Think what data types will you need?*

Exercise #2: More math

- create a program that will ask the user for length of 2 sides of a right-angled triangle
- then calculate the length of the hypotenuse
- Output the length to 2 decimal places
- Bonus: calculate the perimeter and area as well.

Style matters!

- See style guide lines
- (https://bucky.socs.uoguelph.ca/mod/page/view.php?id=888)
- Your assignment must follow these guidelines or you will lose marks!
- This will help your code look more readable!

More Exercises

Go onto bucky and go through

Lesson: Variables and Identifiers &

Lesson: Expressions

In Independent Exercises, Lab 2: Variables and expressions

The only way to get better is to practice!!!

Need Extra help?

- Free tutoring offered by TAs! (Book an appointment on bucky) (also see me after Lab)
- Drop-in help hours (right after your Lecture!) (11:30,2:30 and 5:30) (Tuesday & Thursday)

All meetings will take place in room 001/002 in the basement of Reynolds.

Got a question?

- Ask me! :)
- Post on the Forums: forum.socs.uoguelph.ca
- Email us: cis1500@socs.uoguelph.ca

Protip: Search the forums and bucky before making a post or sending an email!

Or are you Incredibly Bored?

And looking for a Challenge? Or something new?

Come see me after lab!

See bucky for SideQuests!

Side-Quest study group

- Mondays from 2:30-4:00 in 1303 & 1305 Science Complex
- Labs in the hallway behind Second Cup
- Bring your pi and any project ideas you have
- sign up on bucky!

****If you want to come but there is a schedule conflict, Email cis1500@socs.uoguelph.ca so that we can plan more! :)****

Sites To Check To Stay up to Date

Course Website: bucky.socs.uoguelph.ca

SOCS Forums: forum.socs.uoguelph.ca

Textbook: zybooks.com

Reminders!

- **Complete assigned textbook readings** before 9 am Tomorrow(Tuesday)!
- Complete academic integrity quiz on Moodle(see bucky for instructions!)
    - must be completed by October 10th
- **Protip: Finish assigned readings before your Lab!** (The lab will feel like a light breeze if you do that)

CIS*1500 Lab 6

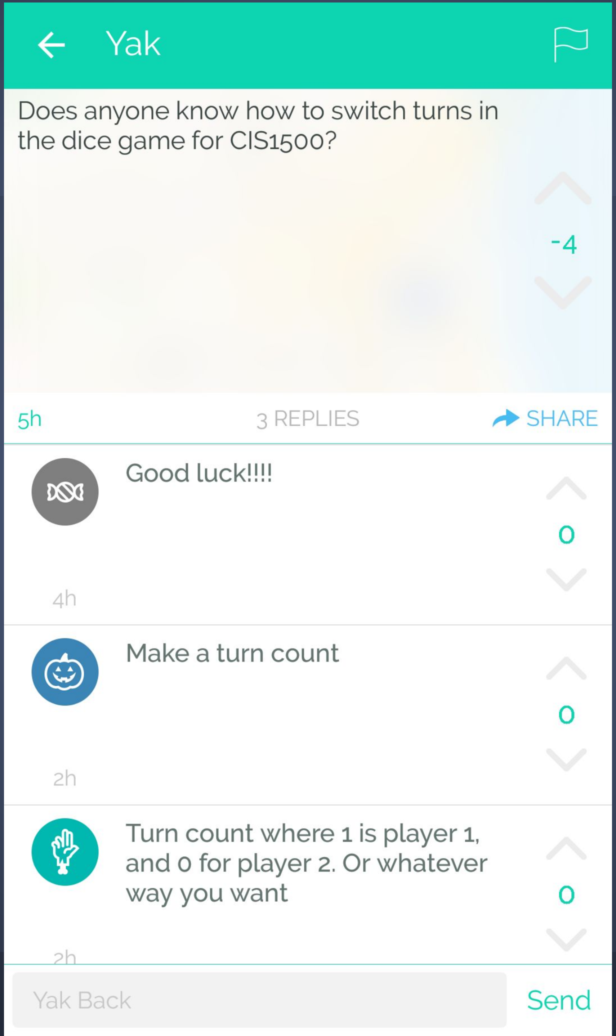
Introduction to Programming
TA: Alliyya Mo

Today’s Lab

- A1 review
- Lab exam things
- StyleChecker!
- Git review
- Lab Exercise
- Git exercise
- Debugging practice

Make sure you book an appointment on Bucky to get your Assignment graded!

Failure to do so will result in a 0!



Does anyone know how to switch turns in the dice game for CIS1500?

-4

5h 3 REPLIES SHARE

Good luck!!!!

0

4h

Make a turn count

0

2h

Turn count where 1 is player 1, and 0 for player 2. Or whatever way you want

0

2h

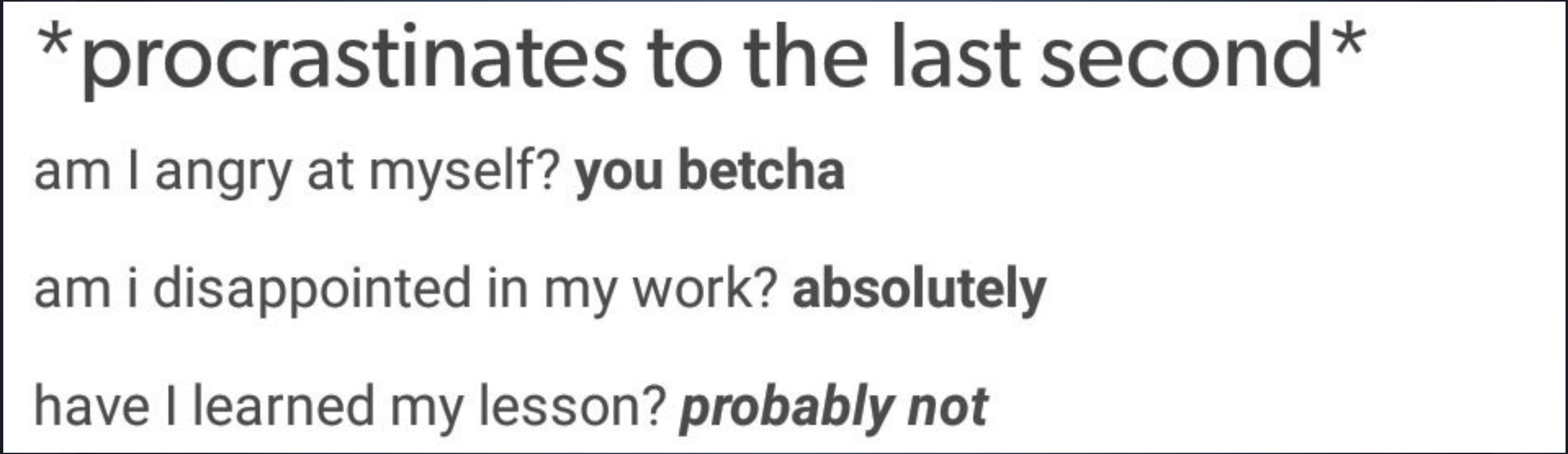
Assignment 1 Grading

1. All grading takes place in Room 001/002 in the Reynolds Building (in the basement). Your grading time will take place sometime during the hour you have selected- be ready to be graded at the beginning of the hour.
2. A TA will grade your assignment.
3. You do not need to bring your pi to the grading. The TA will grade the version of the assignment that you submitted on the course website.

Assignment One Grading Appointments

1. You may book exactly one appointment. Changes to your appointment time must be made at least 8 hours in advance.
2. Failure to book an appointment will result in a grade of zero for the assignment.
3. Failure to keep your appointment results in a 10% grade reduction. You have to rebook the appointment.
4. Failure to keep a second appointment results in an additional 10% grade reduction.
5. Failure to keep a third appointment results in a grade of zero for the assignment.

Learn your lesson!



procrastinates to the last second

am I angry at myself? **you betcha**

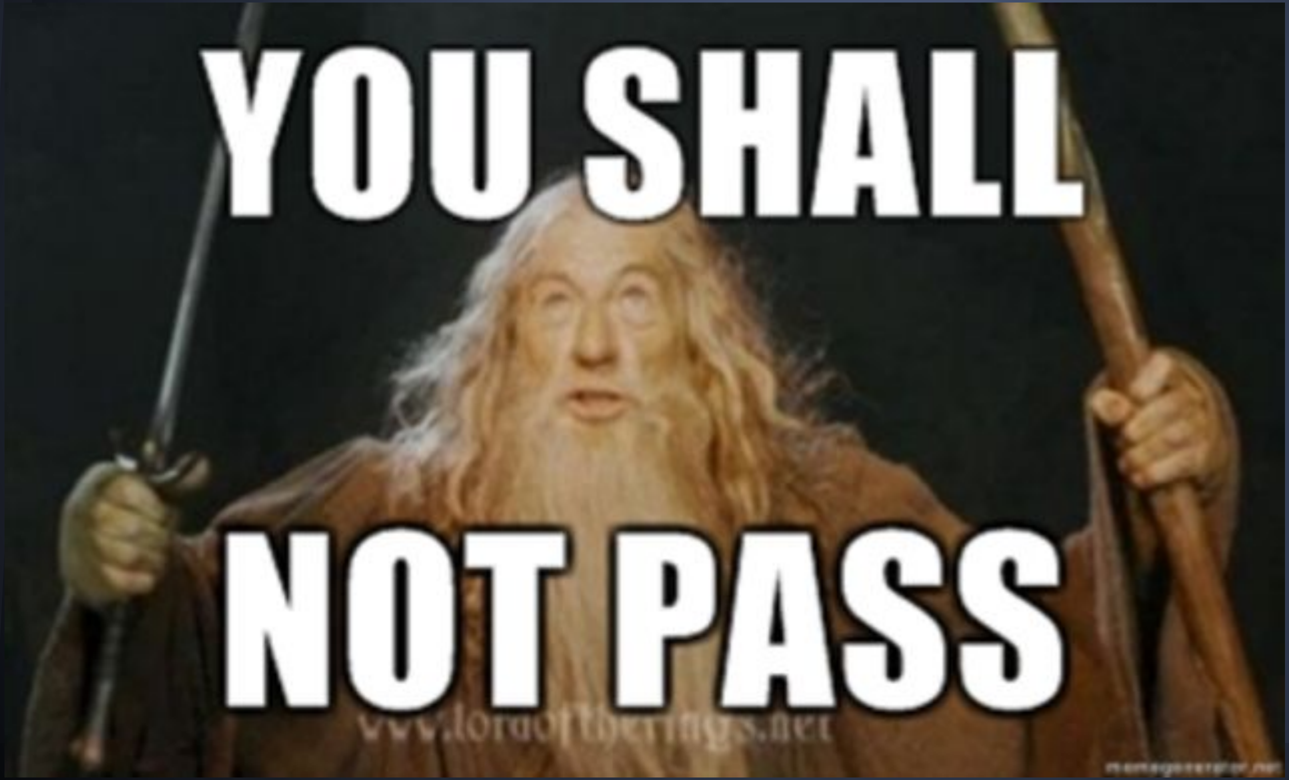
am i disappointed in my work? **absolutely**

have I learned my lesson? ***probably not***

Lab Exam! Next Week!

- Show up on time! (Try to come early)
- You are allowed one cheat sheet
  - letter sized, hand written back and front
- No access to any of your other files
- No Internet(other than Bucky) NO IDES
- No Cell phones/Laptops
- Two TAs will be monitoring you. We see everything!

IF YOU FORGET YOUR PI



REMEMBER:

- HDMI
- POWER CABLE
- Your PI
- Your Goddamn PI!

Warm-up Exercise

Create a program that will print all the even numbers from 0-100

Bonus if you’re quick: Do it backwards from 100 - 0

Submit it to git!

```
git add file.c
git commit -a -m “my message here”
git push
```

In case of fire

1. git add
2. git commit -m “Fire”
3. git push
4. exit building

Stylechecker!

Open up 2 terminals, change both into the directory of your A1

type: styleCheck

and follow its instructions.

Use this tool to fix up the style of your A1 and future assignments! (We're not running style checker for grading this assignment) Good coding style is easy marks! (also makes your life way easier)

Build on previous lab!

Create a basic calculator.(+,-,*/,^,)

Allow the user to select an operation and have them enter the required inputs and then display the answer.

Hint: What data type will be more precise?

Any libraries you might need?

Allow the user to return to the menu after receiving their answer

A Waste of Time?

- create a program that prints out the times tables from 1-12 in a nice grid
- styleCheck it!
- submit to git

Style matters!

- See style guide lines
- (https://bucky.socs.uoguelph.ca/mod/page/view.php?id=888)
- Your assignment must follow these guidelines or you will lose marks!
- This will help your code look more readable!

More Exercises

Go onto bucky and go through

Lesson: Intro to Loops &

Lesson: Kinds of Loops &

Lesson: Program Usability

In Independent Exercises, Lab 6: Loops &

Lab 7: Input and menu-driven interfaces

Need Extra help?

- Free tutoring offered by TAs! (Book an appointment on bucky) (also see me after Lab)
- Drop-in help hours (right after your Lecture!) (11:30,2:30 and 5:30) (Tuesday & Thursday)

All meetings will take place in room 001/002 in the basement of Reynolds.

Got a question?

- Ask me! :)
- Post on the Forums: forum.socs.uoguelph.ca
- Email us: cis1500@socs.uoguelph.ca

Protip: Search the forums and bucky before making a post or sending an email!

Or are you Incredibly Bored?

And looking for a Challenge? Or something new?

Come see me after lab!

Join GCC!

See bucky for SideQuests!(Also email cis*1500)

Sites To Check To Stay up to Date

Course Website: bucky.socs.uoguelph.ca

SOCS Forums: forum.socs.uoguelph.ca

Textbook: zybooks.com

Reminders!

- **Complete assigned textbook readings** before 9 am Tomorrow(Tuesday)!
- Complete academic integrity quiz on Moodle(see bucky for instructions!)
  - must be completed by October 10th
- **Protip: Finish assigned readings before your Lab!** (The lab will feel like a light breeze if you do that)

Final Reminders!

- Book an grading appointment on Bucky for your Assignment!
- REMEMBER YOUR PI for next week
- Protip: Review your loops…
- Also if you don’t remember how to compile, put it on your cheat sheet!
- Don’t miss next week’s lab(Worth same amount as Assignment 1)

CIS*1500 Lab 8

Introduction to Programming
TA: Alliyya Mo

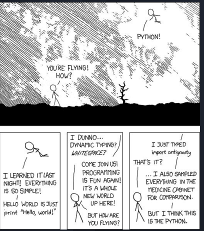
Today’s Lab

- Overview of Functions
- Warm up
- Programming Exercise
- Debugging practice with functions
- Lab exam review/discussion

Check out the Side Quests offered this week!

Side Quest Schedule (In the Thorn Pi Lab)

Workshop:	Date	Time
Git	Wednesday, Nov. 4th	4:30
Extra Practice	Thursday, Nov. 5th	5:30
A1 Review Session	Friday, Nov. 6th	5:30
Python	Monday, Nov. 9th	5:15
Git	Wednesday, Nov. 11th	12:30
Advanced C	Friday, Nov. 13th	5:30
Makefiles and Other Command line tools	Monday, Nov. 16th	5:00


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
...I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

What is a function?

- A group of statements that together perform a task

Return Type

Name of Function

Parameters

```
int squared(int num)
{
    return num*num;
}
```

Function arguments/parameters act as local variables within the function

Pass by Value

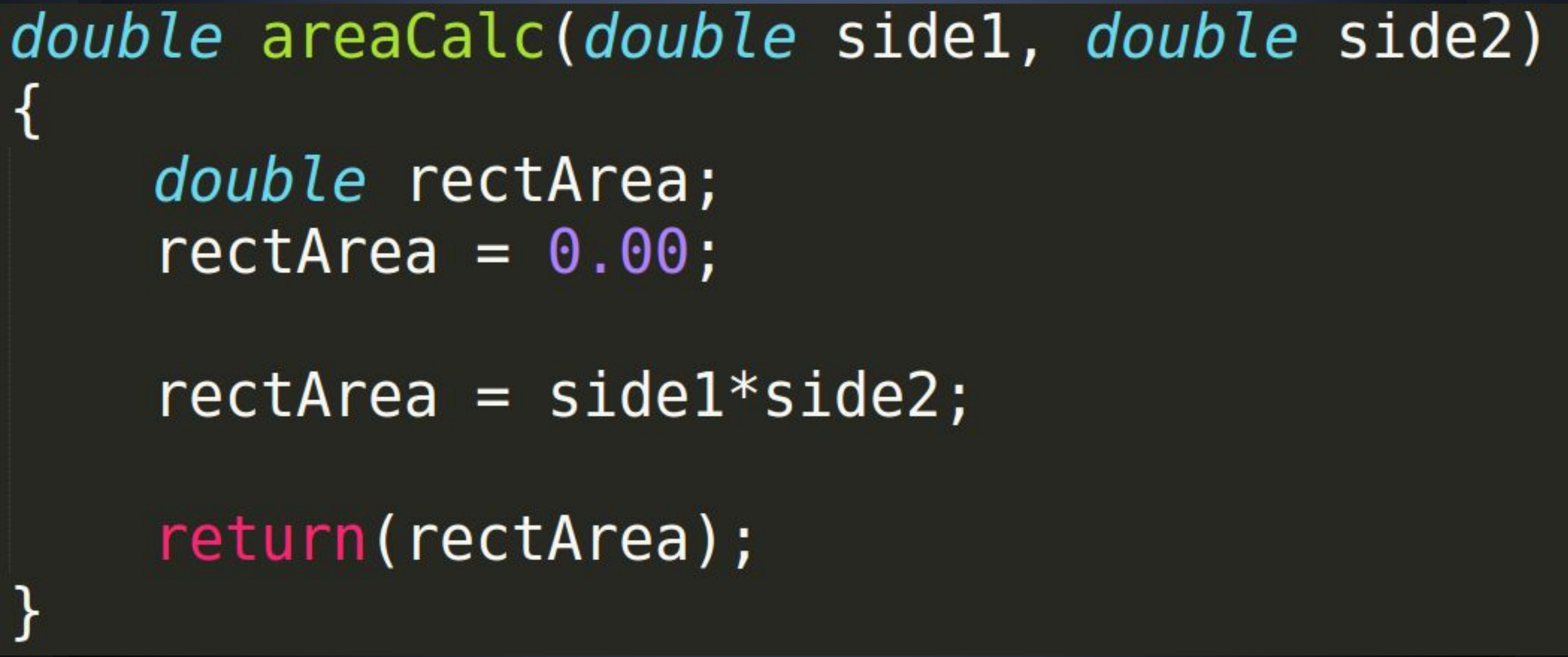
ans = squared(5);

//int squared(int num)

Copies the actual value of the argument(5) into the formal parameter(num) of the function.

The code in the function cannot alter the arguments used to call the function!

A note about scope



```
double areaCalc(double side1, double side2)
{
    double rectArea;
    rectArea = 0.00;

    rectArea = side1*side2;

    return(rectArea);
}
```

Why use functions?

- Readability
- Reusability → You can create your own library of useful functions
- Divide and Conquer
- Allows us to test small parts of our program
- Modularity!
- sheer laziness… 500 loc vs 150
- There’s literally no escape from using them...

Warm-up Exercise

Create a program with the following functions:

areaOfTriangle: will return the area of a triangle given the base and height

printArea: will print out the area with units, given the area of the shape

Submit it to git!

```
git add file.c
git commit -a -m “my message here”
git push
```

Warm-up Exercise

Create a program with the following function:

expoCalc: Calculates exponents given the base and power and returns the result.

<assume positive integers for the base and power>

Submit it to git!

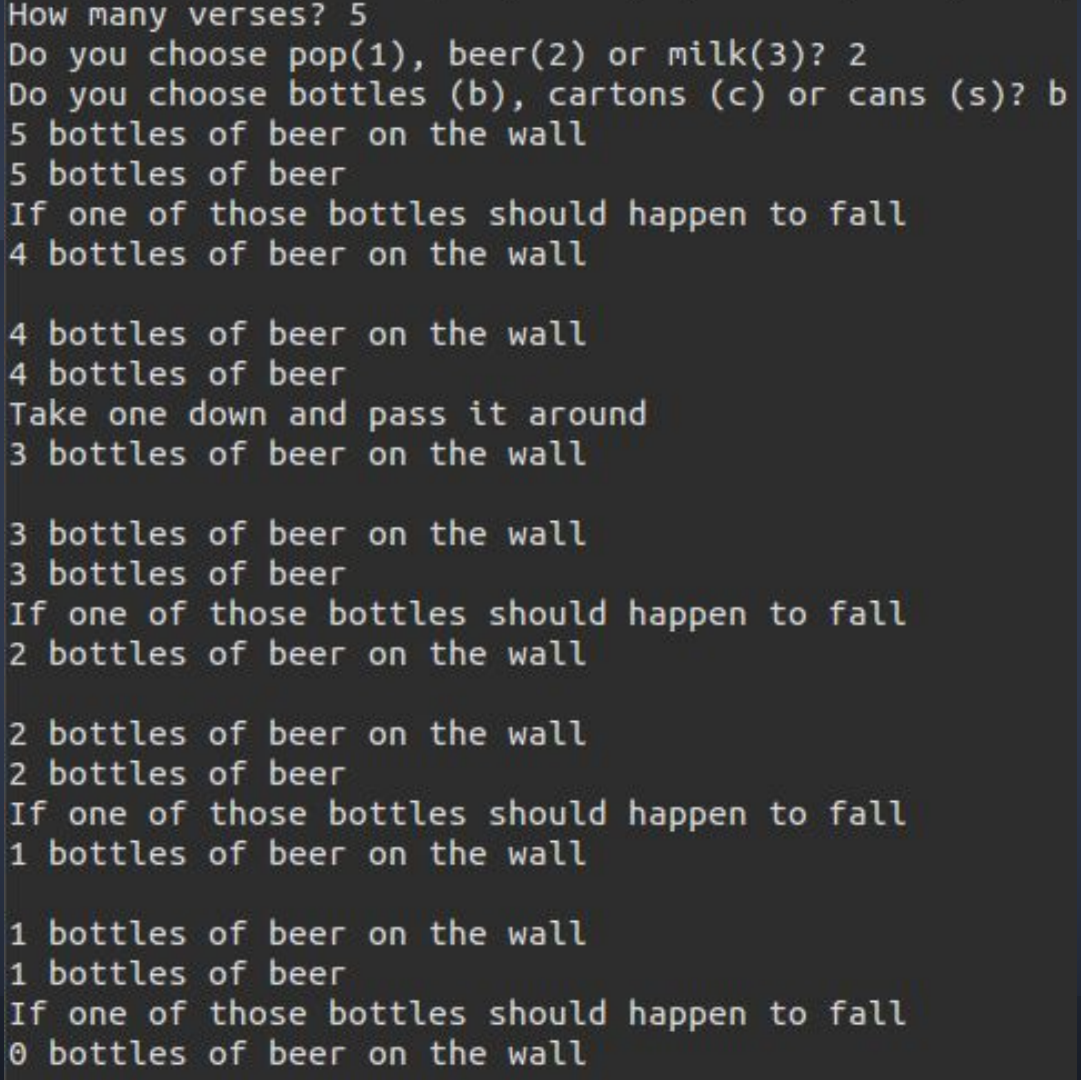
```
git add file.c
git commit -a -m “my message here”
git push
```

Lab Exercise #1: 99 bottles of beer on the wall

- Write a program that prints the lyrics to the song 99 bottles of beer on the wall.
- The program must ask the user how many verses to print and that will be the starting number (i.e. user enters 10 and the song starts at 10 bottles).
- The user must have a choice of beverage type (pop, beer, water) and a choice of container (bottles, cartons, cans).
- The program must randomly determine which third line to print: Take one down and pass it around or If one of those bottles should happen to fall.

Functions we’ll be creating

```
char chooseContainer(void)
int chooseItem(void)
void printContainer(char choice)
void printItem(int bev)
void printThirdLine(char container)
```



```
How many verses? 5
Do you choose pop(1), beer(2) or milk(3)? 2
Do you choose bottles (b), cartons (c) or cans (s)? b
5 bottles of beer on the wall
5 bottles of beer
If one of those bottles should happen to fall
4 bottles of beer on the wall

4 bottles of beer on the wall
4 bottles of beer
Take one down and pass it around
3 bottles of beer on the wall

3 bottles of beer on the wall
3 bottles of beer
If one of those bottles should happen to fall
2 bottles of beer on the wall

2 bottles of beer on the wall
2 bottles of beer
If one of those bottles should happen to fall
1 bottles of beer on the wall

1 bottles of beer on the wall
1 bottles of beer
If one of those bottles should happen to fall
0 bottles of beer on the wall
```

Common Lab Exam Mistakes

- scanf(%d, var);
- #include <stdio.h> // forgetting this
- How to use a loop... or an if statement..

```
while(var < 10);
{
    var--;
}
```

```
for(i--);
{
    something();
}
```

```
if(cond);
{
    something();
}
```

More Exercises

Under Third Quarter:

Go onto bucky and go through

Lesson: Functions

In Independent Exercises,

Lab 8: Functions

The only way to get better is to practice!!!

Got a question?

- Ask me! :)
- Post on the Forums: forum.socs.uoguelph.ca
- Email us: cis1500@socs.uoguelph.ca

Protip: Search the forums and bucky before making a post or sending an email!

Need Extra help?

- Free tutoring offered by TAs! (Book an appointment on bucky) (also see me after Lab)
- Drop-in help hours (right after your Lecture!) (11:30,2:30 and 5:30) (Tuesday & Thursday)

All meetings will take place in room 001/002 in the basement of Reynolds.

Or are you Incredibly Bored?

And looking for a Challenge? Or something new?

Come see me after lab!

Join GCC!

See bucky for SideQuests!(Also email cis*1500)

Sites To Check To Stay up to Date

Course Website: bucky.socs.uoguelph.ca

SOCS Forums: forum.socs.uoguelph.ca

Textbook: zybooks.com

Reminders!

- **Complete assigned textbook readings** before 9 am Tomorrow(Tuesday)!
- **Protip: Finish assigned readings before your Lab!** (The lab will feel like a light breeze if you do that)
- Get started on your assignment!

CIS*1500 Lab 9

Introduction to Programming
TA: Alliyya Mo

Today's Lab

- Review of Functions
- Function Prototypes
- Arrays
- Warm up
- Programming Exercise 1
- Programming Exercise 2

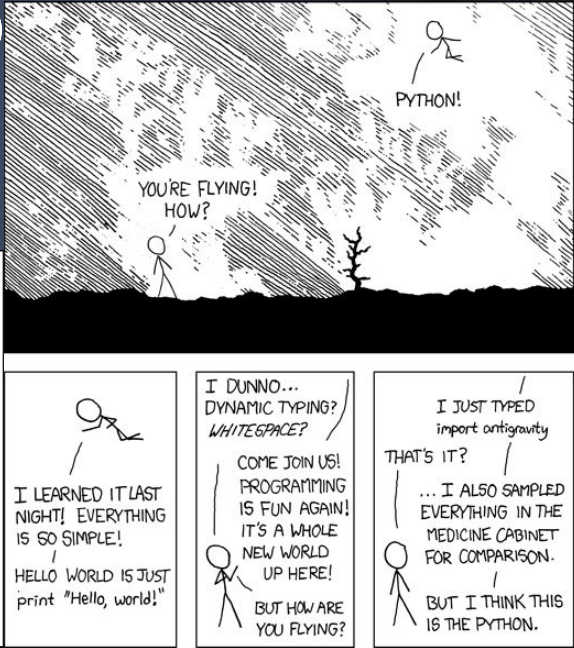
Check out the Side Quests offered this week!

Side Quest Schedule (In the Thorn Pi Lab)

Workshop:	Date	Time
Python	Monday, Nov. 9th	5:15
Git	Wednesday, Nov. 11th	12:30
Advanced C	Friday, Nov. 13th	5:30
Makefiles and Other Command line tools	Monday, Nov. 16th	5:00

COMMENT
DATE
CREATED MAIN LOOP & TIMING CONTROL — 14 HOURS AGO
ENABLED CONFIG FILE PARSING — 9 HOURS AGO
MISC BUGFIXES — 5 HOURS AGO
CODE ADDITIONS/EDITS — 4 HOURS AGO
MORE CODE — 4 HOURS AGO
HERE HAVE CODE — 4 HOURS AGO
AAAAAAAA — 3 HOURS AGO
ADKFJSLKDFJSDKLFJ — 3 HOURS AGO
MY HANDS ARE TYPING WORDS — 2 HOURS AGO
HAAAAAAAAANDS — 2 HOURS AGO
AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

MAN, I SUCK AT THIS GAME. CAN YOU GIVE ME A FEW POINTERS?
0x3A28213A 0x6339392C, 0x7363682E.
I HATE YOU.

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
...I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

What is a function?

- A group of statements that together perform a specific task

Return Type

Name of Function

Parameters

```
int squared(int num)
{
    return num*num;
}
```

Function def’n:

Its name

Its input

Its output

How it works

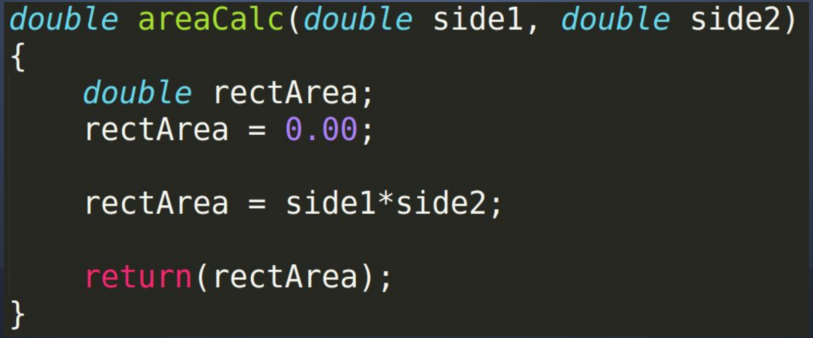
Function Definition

Name: Name of Function(**areaCalc)**

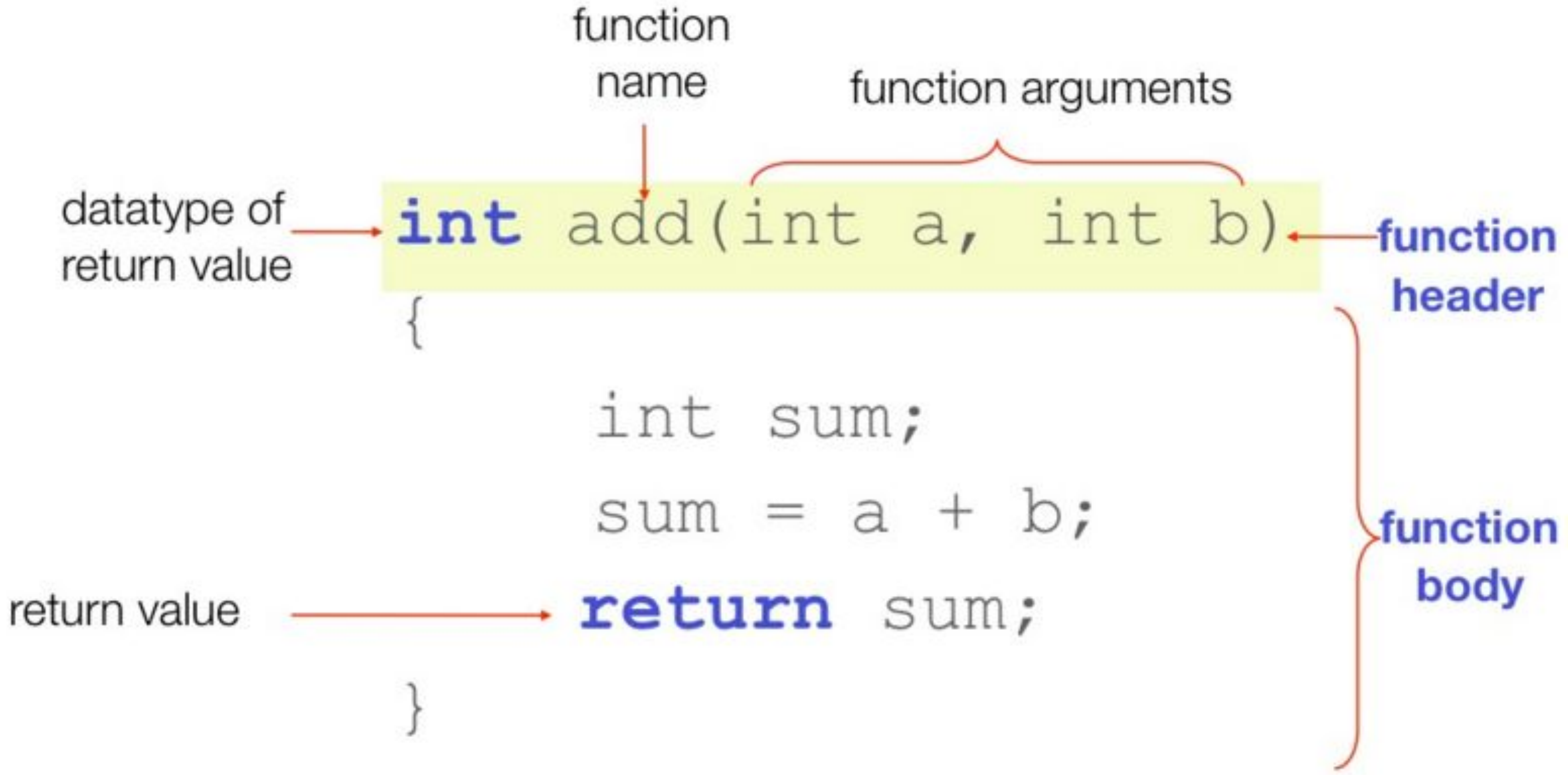
Input:Parameters (**double side1, double side2**)

Output: Return Value (rectArea) (**double**)

How it works: Body of Function



```
double areaCalc(double side1, double side2)
{
    double rectArea;
    rectArea = 0.00;

    rectArea = side1*side2;

    return(rectArea);
}
```



```
int add(int a, int b)
{
        int sum;
        sum = a + b;
       return sum;
}
```

Calling a Function

- We **call** other functions from within main or within functions called from main.
- Functions can call other functions, and can even call themselves (**recursion**).
- When a function is called, execution of main stops, the function is
- executed, then execution of main resumes (or execution of the calling program/function)
-

Pass by Value

```
ans = squared(5);
//int squared(int num)
```

Copies the actual value of the argument(5) into the formal parameter(num) of the function.

The code in the function cannot alter the arguments used to call the function!

Pass by Reference

```
void squared(int *num)
{
    num= num*num;
}
```

```
int main(void)
{
        int number;
        number = 10;
        squared(&number);
        return(0);
}
```

This means that the variable’s address in memory is passed to the function. As such, the variable you access in the function is not a copy of your variable in main, but instead the variable itself.

Function Prototype

Functions must be declared before they are called in the main function!

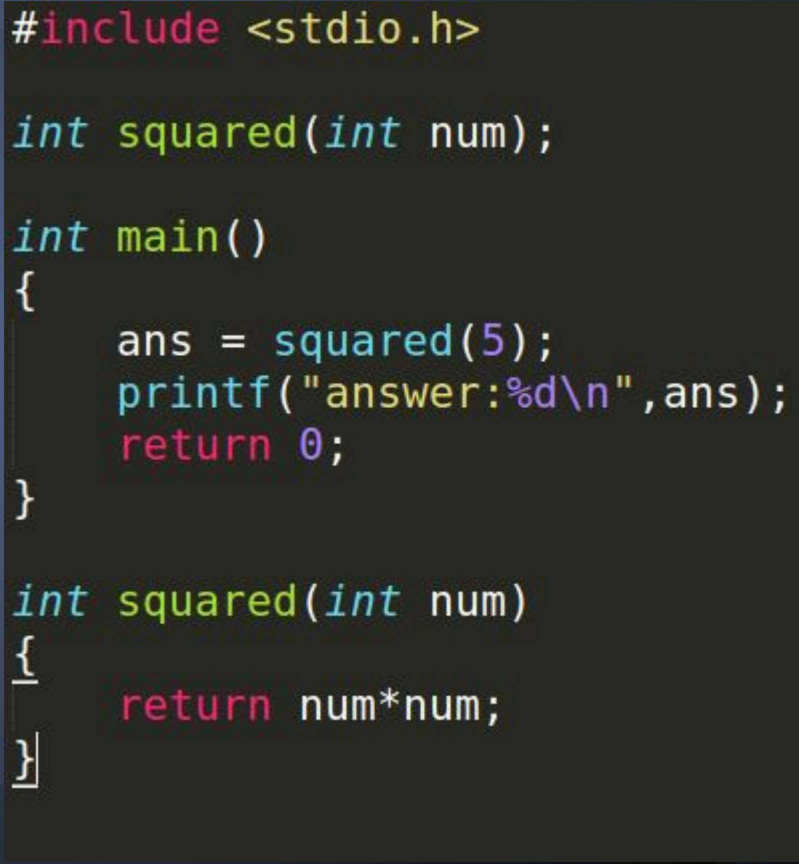
- put the whole function before main(how we’ve been doing it before)
- put a **function prototype** before main!

Function Prototype

includes:

- the returntype,
- name of the function
- the type of the parameters

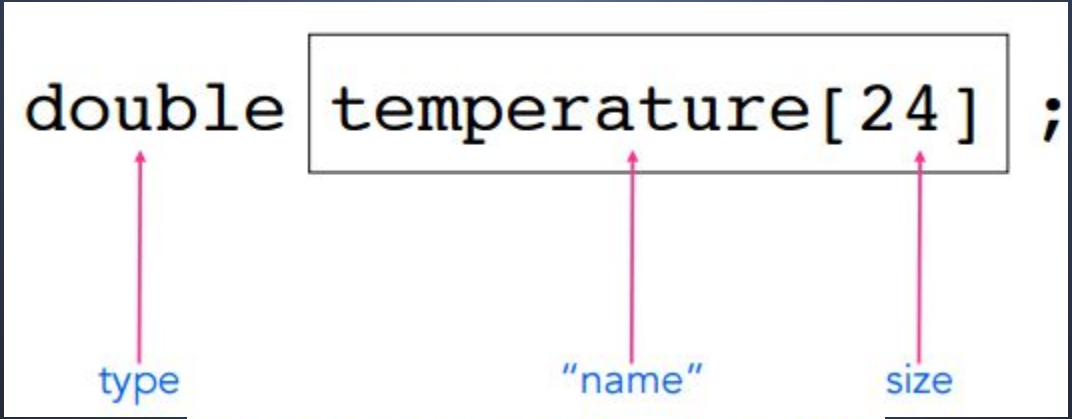
(basically the first line of our function def’n)



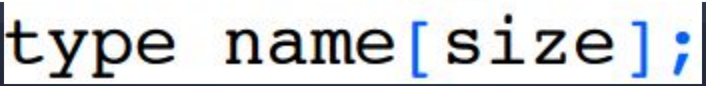
```
#include <stdio.h>

int squared(int num);

int main()
{
    ans = squared(5);
    printf("answer:%d\n",ans);
    return 0;
}

int squared(int num)
{
    return num*num;
}
```

Arrays

An array is a consecutive series of variables that share one variable name.



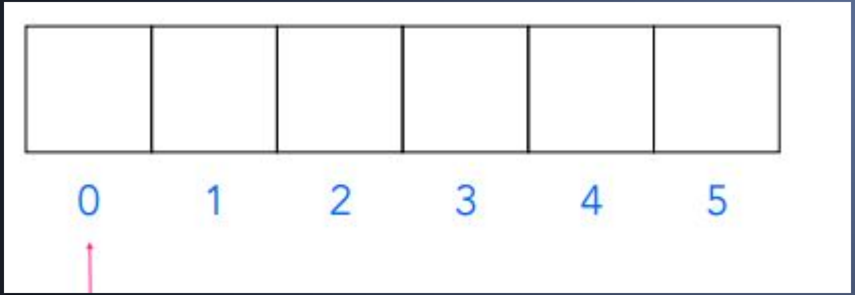
```
double temperature[24] ;
```



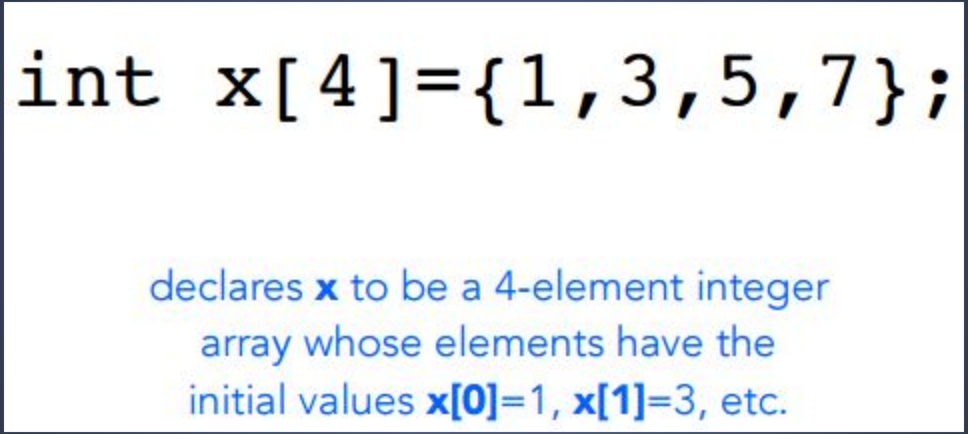
```
type name[size];
```

Arrays start at 0

int numArray[6]   //6 elements



Indexed from 0-5



```
int x[4]={1,3,5,7};
```

declares **x** to be a 4-element integer array whose elements have the initial values **x[0]**=1, **x[1]**=3, etc.

Array elements

loop indexes as
array subscripts

```
int i, z[100];

for (i=0; i<100; i=i+1)
     z[i] = i;
```

each time the value
of a control variable
is incremented, the
next array element is
selected.

for loop to process
elements in an array

How many elements?

So far, we’ve been allocating arrays statically. This means that we cannot increase or reduce the size of our array after it has been declared.

It is possible to create dynamic arrays, but this won’t be covered until CIS*2500.

As a rule of thumb, when using static arrays is it better to have too much room as opposed to too little.

Probably not going to want to do arrayName[9999999]

?

Options:

Declare the array statically with a large usuable size, ex. 50 - 100.(preferred for CIS*1500)

Figure out the size of the array then declare it with a variable size (feature of C99 standard)

Dynamic arrays!(CIS*2500)

Warm-up Exercise (don’t For-get anything)

Create a program that does the following:

- create 2 character arrays, one with 10 elements and one to hold your name.
- fill the other character array with user input
- print out the characters of your name one line at a time
- print it backwards
- print the other array all in one line

Submit it to git!

```
git add file.c
git commit -a -m “my message here”
git push
```

Lab Exercise #1

Create a program with the following functions:

Ask the user how many characters they’d like to input.

Have the user enter their characters and save those characters into an array. then create these functions using pass by reference

replace() → will replace every vowel with a ‘?’

printCharArray() → will print out the character array
must use function prototypes

Submit it to git!

Lab Exercise #2: Class average

Create a program that asks the user, how many grades they'd like to input.

create an array to hold these grades

create the following functions:

setGrades()   // get the user to enter a grade for each element of the array

classAverage() //calculates the average all the grades

printGrades() //will print out the grades for each student

must use function prototypes

More Exercises

Under Third Quarter:

Go onto bucky and go through

Lesson: Functions

In Independent Exercises,

Lab 8: Functions

The only way to get better is to practice!!!

Got a question?

- Ask me! :)
- Post on the Forums: forum.socs.uoguelph.ca
- Email us: cis1500@socs.uoguelph.ca

Protip: Search the forums and bucky before making a post or sending an email!

Need Extra help?

- Free tutoring offered by TAs! (Book an appointment on bucky) (also see me after Lab)
- Drop-in help hours (right after your Lecture!) (11:30,2:30 and 5:30) (Tuesday & Thursday)

All meetings will take place in room 001/002 in the basement of Reynolds.

Or are you Incredibly Bored?

And looking for a Challenge? Or something new?

Come see me after lab!

Join GCC!

See bucky for SideQuests!(Also email cis*1500)

Sites To Check To Stay up to Date

Course Website: bucky.socs.uoguelph.ca

SOCS Forums: forum.socs.uoguelph.ca

Textbook: zybooks.com

Reminders!

- **Complete assigned textbook readings** before 9 am Tomorrow(Tuesday)!
- **Protip: Finish assigned readings before your Lab!** (The lab will feel like a light breeze if you do that)
- Get started on your assignment!

CIS*1500 Lab 11

Introduction to Programming
TA: Alliyya Mo

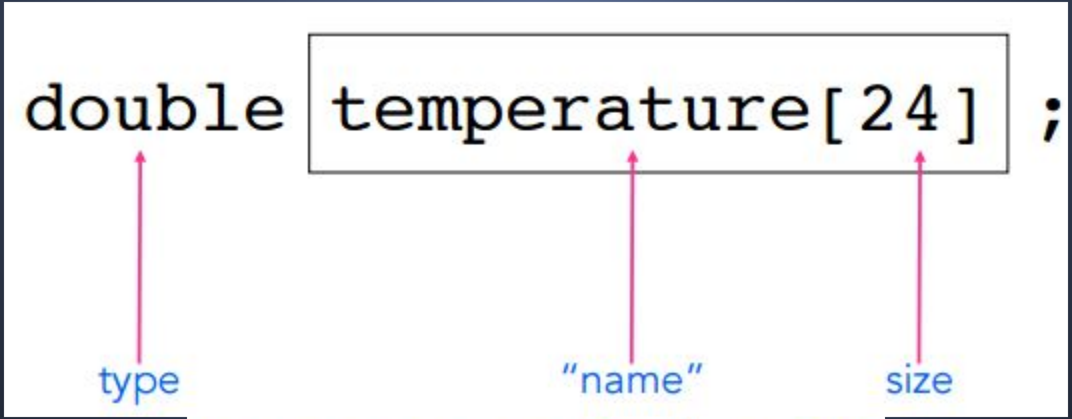
Today’s Lab

- Overview of arrays
- Review passing arrays to functions
- **ctype** library (character functions)
- using **fgets**
- **string** library (string functions)
- reading from a file using **fgets**

Assignment 2 due on Sunday, November 29th 11:55pm!

Arrays

An array is a consecutive series of variables that share one variable name.

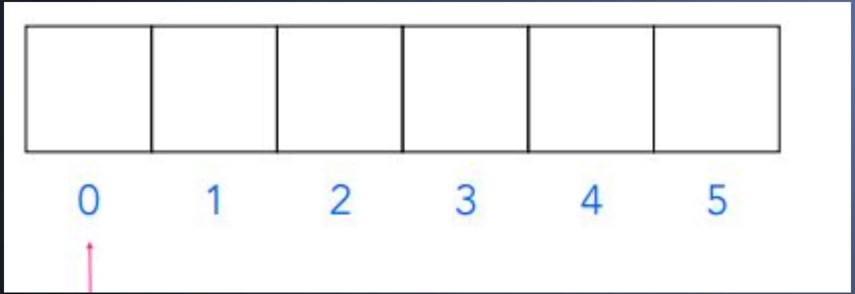


```
double temperature[24] ;
```

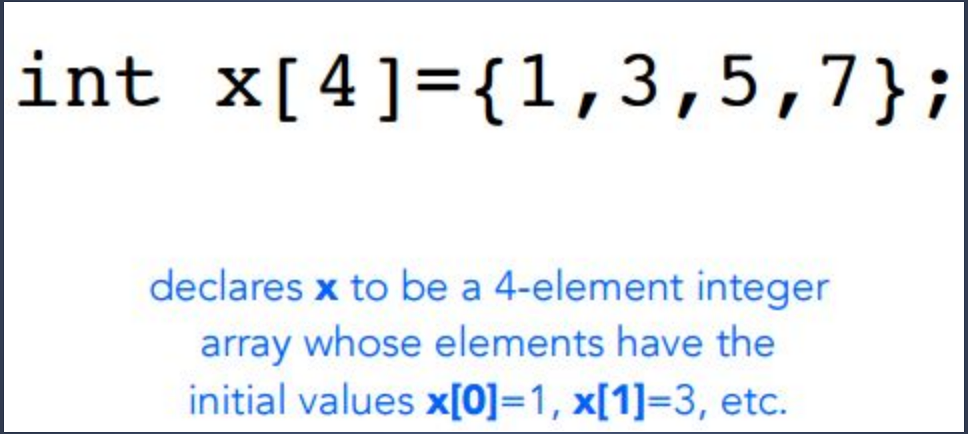
```
type name[size];
```

Arrays start at 0

int numArray[6]   //6 elements



Indexed from 0-5



```
int x[4]={1,3,5,7};
```

declares **x** to be a 4-element integer array whose elements have the initial values **x[0]**=1, **x[1]**=3, etc.

Array elements

loop indexes as
array subscripts

```
int i, z[100];

for (i=0; i<100; i=i+1)
     z[i] = i;
```

each time the value
of a control variable
is incremented, the
next array element is
selected.

for loop to process
elements in an array

What size to make my array???

Options:

Declare the array statically with a large usuable size, ex. 50 - 100.(preferred for CIS*1500) ex. char array[100];

Figure out the size of the array then declare it with a variable size (feature of C99 standard)

Dynamic arrays!(CIS*2500)

How many elements?

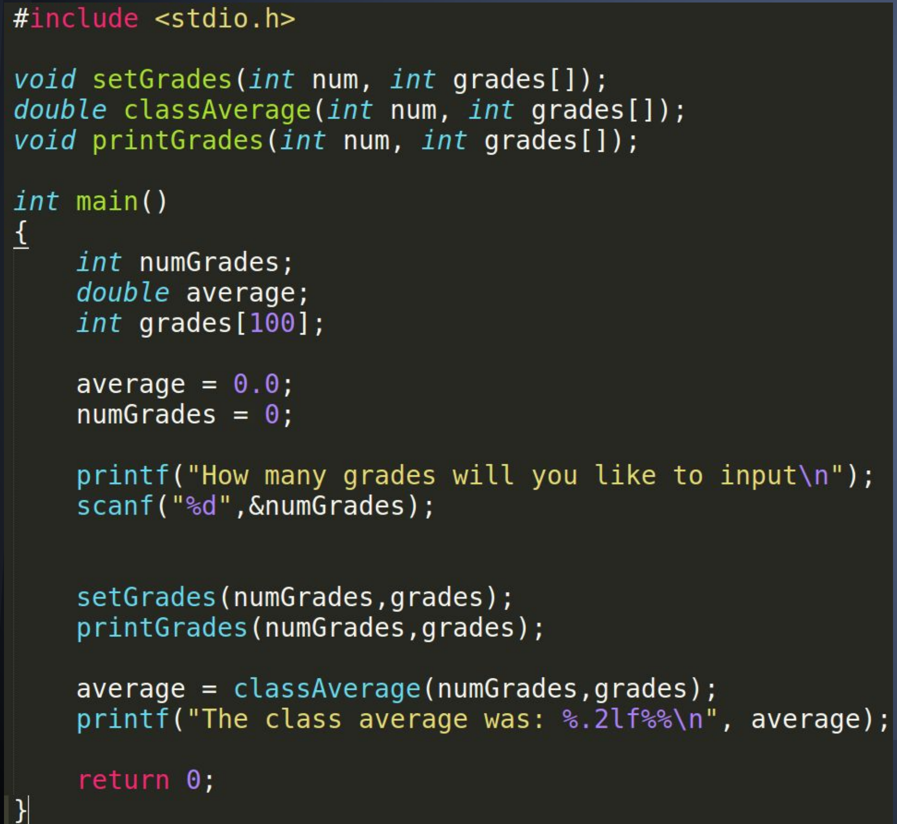
So far, we've been allocating arrays statically. This means that we cannot increase or reduce the size of our array after it has been declared.

It is possible to create dynamic arrays, but this won't be covered until CIS*2500.

As a rule of thumb, when using static arrays is it better to have too much room as opposed to too little.

Probably not going to want to do arrayName[9999999]

Passing arrays to functions



```c
#include <stdio.h>

void setGrades(int num, int grades[]);
double classAverage(int num, int grades[]);
void printGrades(int num, int grades[]);

int main()
{
    int numGrades;
    double average;
    int grades[100];

    average = 0.0;
    numGrades = 0;

    printf("How many grades will you like to input\n");
    scanf("%d",&numGrades);


    setGrades(numGrades,grades);
    printGrades(numGrades,grades);

    average = classAverage(numGrades,grades);
    printf("The class average was: %.2lf%%\n", average);

    return 0;
}
```

Quick Warm-up Exercise

Create a program that declares a character array of any size, fill it with your name.

Create a function that prints out your name.

call this function

Submit it to git!

```
git add file.c
git commit -a -m “my message here”
git push
```

ctype.h

- a library containing character functions
  - including: (man ctype.h for more)
    - int isalpha(int c)
    - int islower(int c)
    - int isupper(int c)
    - int isspace(int c)
    - int tolower(int c)
    - int toupper(int c)

- **int isalpha(int c)**
  - Returns non-zero if c is an alphabet character, 0 otherwise.
- **int isupper(int c)**
  - Returns non-zero if c is an uppercase letter, 0 otherwise.
- **int islower(int c)**
  - Returns a non-zero if c is a lowercase letter, 0 otherwise
- **int isdigit(int c)**
  - Returns non-zero if c is a digit (0, 1, ..., 9), 0 otherwise.

Warm-up Exercise

Using the ctype library(include it just how we included the math library and the standard library)

Read in a character array of 20 characters, then loop through the array checking if the character is lowerCase, if it is convert it to uppercase.

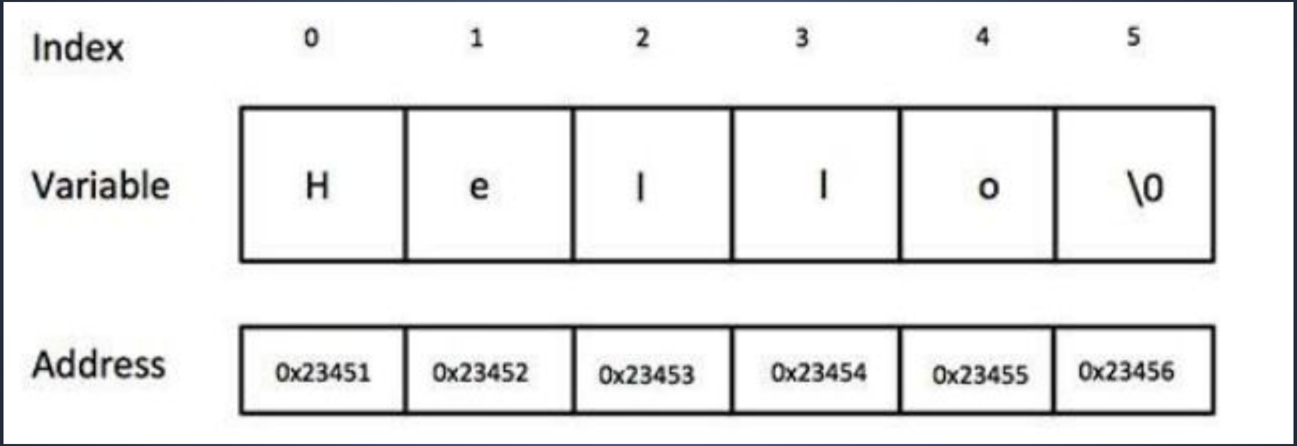
Then print out the character array

Use and create any functions you feel fit!

Strings!

- strings are just one dimensional arrays of characters with a special ending character
  - the null terminator ‘\0’
- The final character of the array must be a null terminator in order for it to be a string to perform string based functions upon it
- You must size your arrays keeping in mind of this null terminator

char hello[] = “hello”;



Index	0	1	2	3	4	5
Variable	H	e	l	l	o	\0
Address	0x23451	0x23452	0x23453	0x23454	0x23455	0x23456

Null Terminator (‘\0’)

- The null terminator is a special character that in a character indicates the end of the string
- For a character array to be considered a valid string it must have a null terminator

Format Specifier for Strings

Since these character arrays are strings, we no longer have to print the character array one element at a time

We can use the format specifier %s in our print statements

ex. printf(“Our string %s”,hello);

string.h

- The string library contains dozens of functions that can be used to manipulate and work with strings including:
    - Finding the length of a string
    - Copying one string to another
    - Concatenating strings
    - Comparing 2 strings

Functions in the Library

- This library includes
  - char ***strcat**(char *restrict, const char *restrict);
  - int **strcmp**(const char *, const char *);
  - char ***strcpy**(char *restrict, const char *restrict);
  - size_t **strlen**(const char *);
  - char ***strtok**(char *restrict, const char *restrict);

  We’ll be using strlen and strcpy today.

  Feel free to experiment using other string functions

strlen

- The function strlen returns the number of characters in the string not including the null terminator.
- The format of strlen is:

size_t strlen(const char *str)

- **str** – The string whose length is to be found
- **size_t** – Is an unsigned integer type meaning it cannot represent negative values.

On success the function returns the length of the string.

strcpy

- The function strcpy will allow you to copy the contents of one string into another.
- The format of strcpy is:

  char * strcpy(char * dest, const char * src)

- **dest** – The pointer t the destination array where the content is to be copied.
- **src** – The string to be copied.

On success returns a pointer to the destination string **dest.**

strcat

- The function strcat is used to concatenate strings together. That is, it is used to insert the contents of one string to the end of the contents of another string. Also overwriting the null terminator ‘\0’ of the first string.
- The format of strcpy is:

  char *strcat(char *dest, const char *src)

- **dest -** The pointer to the destination array, which should contain a C string, and should be large enough to contain the concatenated resulting string
- **src** – The string to be appended. This should not overlap the destination.
- On success, the function returns a pointer to the resulting string dest

More String Functions from stdlib

int atoi(const char *str)

- Converts the string argument str to an integer (type int).

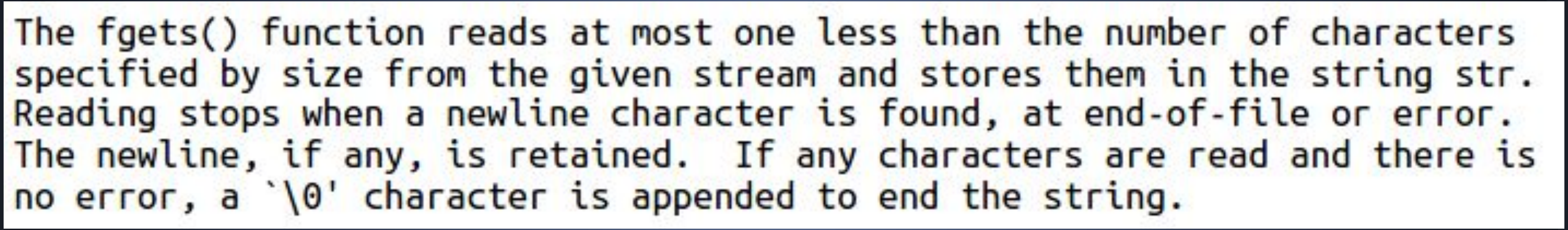
long int strtol(const char *str, char **endptr, int base)

- Converts the initial part of the string in str to a long int value according
- to the given base, which must be between 2 and 36 inclusive, or be the
- special value 0.
- Use base = 10 for regular numbers.

fgets

The answer to life the universe and everything

char *fgets(char *s, int size, FILE *stream)



```
The fgets() function reads at most one less than the number of characters
specified by size from the given stream and stores them in the string str.
Reading stops when a newline character is found, at end-of-file or error.
The newline, if any, is retained.  If any characters are read and there is
no error, a `\0' character is appended to end the string.
```

It return NULL on failure

Standard Streams include stdin, stdout,stderr

fgets

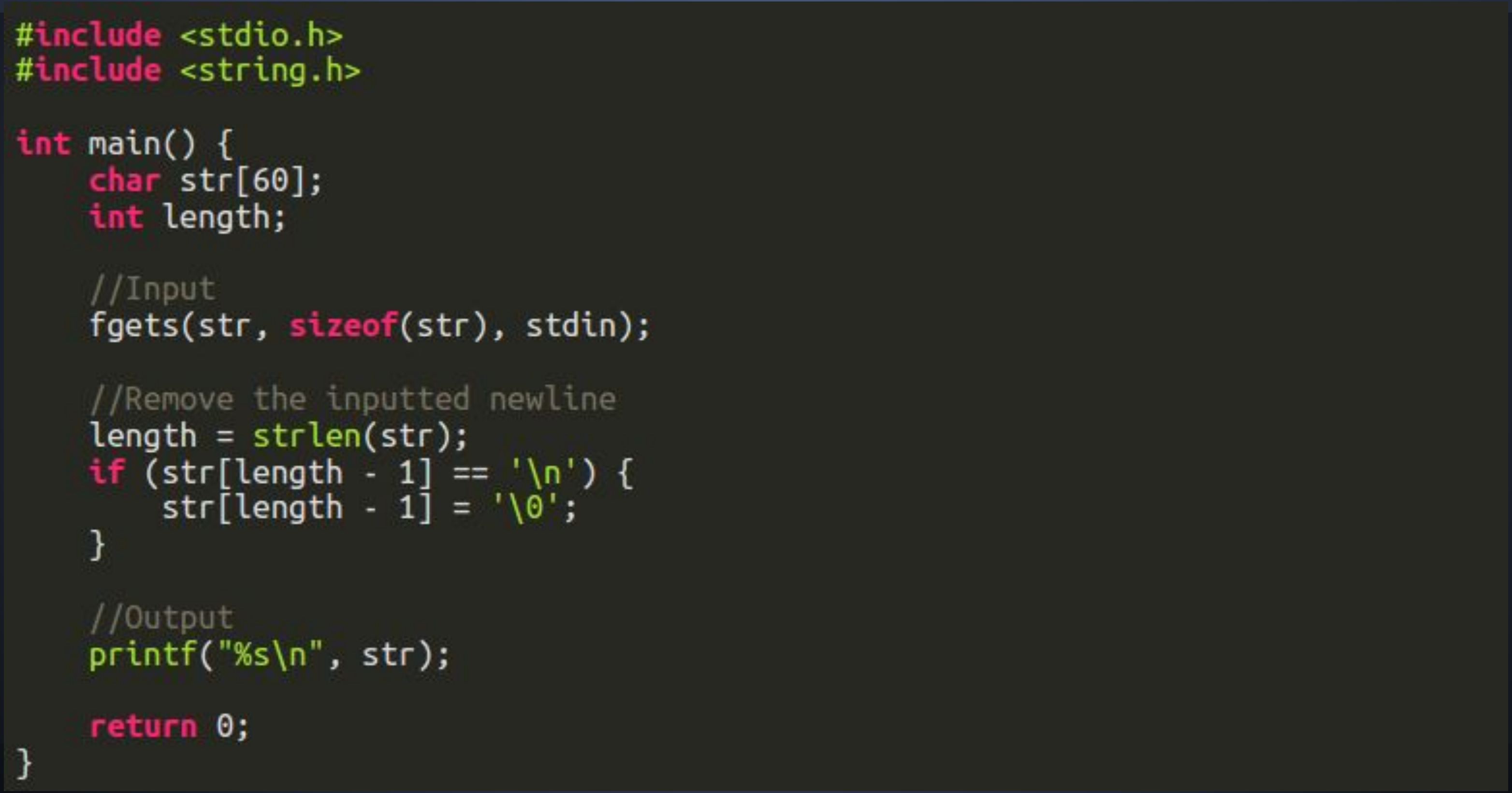
- The format of fgets is:

  char * fgets(char * str, max_char, input_stream);

- **str** – The pointer to an array of characters where the string read is stored.
- **max_chars** – The maximum number of characters to be read (including the null terminator).
- **input_stream** – The pointer to a FILE pointer that identifies the stream where characters are read from.

On success, **fgets** returns the same str parameter. Otherwise, if the End-of-File is encountered and no characters have been read, the contents of str remain unchanged and a null pointer is returned

fgets: comes with a newline



```c
#include <stdio.h>
#include <string.h>

int main() {
    char str[60];
    int length;

    //Input
    fgets(str, sizeof(str), stdin);

    //Remove the inputted newline
    length = strlen(str);
    if (str[length - 1] == '\n') {
        str[length - 1] = '\0';
    }

    //Output
    printf("%s\n", str);

    return 0;
}
```

String exercise!

Get a string from the user(remember to replace the newline with the null terminator)

Make a copy of this string into another string

Then replace every lowercase letter with a 9, and print out both strings

printf(“%s”,string);

More Exercises

Under Third Quarter:

Go onto bucky and go through

Lesson: Functions

In Independent Exercises,

Lab 8: Functions

The only way to get better is to practice!!!

Got a question?

- Ask me! :)
- Post on the Forums: forum.socs.uoguelph.ca
- Email us: cis1500@socs.uoguelph.ca

Protip: Search the forums and bucky before making a post or sending an email!

Need Extra help?

- Free tutoring offered by TAs! (Book an appointment on bucky) (also see me after Lab)
- Drop-in help hours (right after your Lecture!) (11:30,2:30 and 5:30) (Tuesday & Thursday)

All meetings will take place in room 001/002 in the basement of Reynolds.

Or are you Incredibly Bored?

And looking for a Challenge? Or something new?

Come see me after lab!

Join GCC!

See bucky for SideQuests!(Also email cis*1500)

Sites To Check To Stay up to Date

Course Website: bucky.socs.uoguelph.ca

SOCS Forums: forum.socs.uoguelph.ca

Textbook: zybooks.com

Reminders!

- **Complete assigned textbook readings** before 9 am Tomorrow(Tuesday)!
- **Protip: Finish assigned readings before your Lab!** (The lab will feel like a light breeze if you do that)
- Get started on your assignment!

CIS*1500 Review Session!

Introduction to Programming
TA: Alliyya Mo

Today’s Options:

- Q & A, What do you guys what to review
- Going through the whole course
- Going through the review questions from class
- Sample questions from bucky
- Tracing code examples

Things to make sure you know!

- Basic syntax
- Control Flow(decision making and loops)
- Functions (Pass by value and pass by reference)
- Arrays!
- Strings
- **Tracing code(can you read code)***

Basics of C

- Different data types(char,int,float,double)
- printf and format specifiers(%d,%f,%lf)
- using the math library
- How to compile(gcc)
  - flags for compilation(-Wall)

Which is the correct print statement?

Given:

```
int i = 45;
double j = 1.69;
char k = 'A';
```

Output: 45 1.69 A

A. printf(“%d %f %c”,i,j,k);
B. printf(“%d %f %d”,i,j,k);
C. printf(“%d %d %c”,i,j,k);
D. printf(“%d %lf %c”,i,j,k);

How do we include the math library to use in our programs?

A. #include mathlib
B. #include math.h and use the flag -lm when compiling
C. #include mathlib and use the flag -lm when compiling
D. #include <math.h> and use the flag -lm when compiling
E. #include math.h and use the flag -Wall when compiling

Which operation does not results in 0

A. 16%2
B. 16%4
C. 16%10
D. 16%8

Trigonometric function in the math library use ?

A. Degrees
B. Gradients
C. There's trig functions in C?????
D. Radians
E. Moles
F. Gradian
G. Turns

Which the correct use of Pow?

```
int power = 3;
double powerD = 3.0;
int base = 2;
double baseD = 2.0;
int ans;
double ansD;
```

a. ans = pow(power, base);
b. ans = pow(base, power);
c. ansD = pow(power, base);
d. ansD = pow(baseD, powerD);
e. ansD = pow(powerD, baseD);
f. ansD = powd(baseD, powerD);

What -Wall flag do when used in compiling?

Ex. gcc -Wall myFile.c

A. Renames your executable
B. Checks for logic errors
C. Allows you to use the math library
D. Reports back warning messages from compilation
E. Checks that your program is correct

Which is the most correct?

A. #defn PI 3.1456
B. define PI = 3.1456
C. #define PI = 3.1456
D. define PI 3.1456
E. #define PI 3.1456

Control Flow

- if-statement syntax
- switch statements syntax
- for loops
- while loops
- do..while loops

- Common errors with loops(debugging)

What is the results of this?

```
int i;
for (i = 0; i <= 10; i++);
{
    printf("Hello\n");
}
```

a. prints hello 10 times
b. prints hello 11 times
c. prints hello 1 time
d. Doesn’t print anything
e. Segmentation fault

Functions

- How to declare functions
    - What is the return type, parameters
    - Function prototypes
- Pass by Value vs Pass by Reference
- How do we pass an array?

Which statement is true about pass by value?

a. it makes a copy of the variable in the parameters, so nothing in memory changes
b. is only for arrays
c. doesn’t work for arrays
d. same things as pass by reference
e. passes the memory address, so values in memory actually change

Which statement is true about pass by reference

a. it makes a copy of the variable in the parameters, so nothing in memory changes
b. is only for arrays
c. doesn’t work for arrays
d. fancy way of saying pass by value
e. passes the memory address, so values in memory actually change

Arrays

- How to declare an array
- How to transverse an array
- Array indexing
- How to pass an array to a function

Strings

- How are strings defined in C, what’s the difference between a string and a character array?
- The string library includes these functions:
  - strcpy*
  - strcmp
  - strlen

  Make sure you know how to use these functions

Which is the correct usage of strcpy

char name[] = {“Oliver Queen”};

char otherName1[14];

char otherName2[10];

a. strcpy(name, otherName);
b. strcpy(&otherName1, &name);
c. strcpy(otherName2, name);
d. strcpy(otherName1, name);

Structs

- What is a struct
- Know how to declare these
- How to access parts of a struct

Is this a valid declaration of a struct?

```
struct Student
{
    int id;
    int age;
    int average;
    char firstName[10];
    char lastName[10];
};
```

Testing

- What are the different types of testing?
- How do they work?

Which are not a type of Testing?

a. Yolo testing
b. Black box testing
c. White box testing
d. Grey box testing
e. Unit testing
f. Regression testing

Suggestions!

- Practice reading code! Look over a friend’s piece of code, can you understand what it’s trying to do?
- Would you be able to fill in blanks in a section of code
- If you don’t know how a certain function works, write some code! Practice!