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!

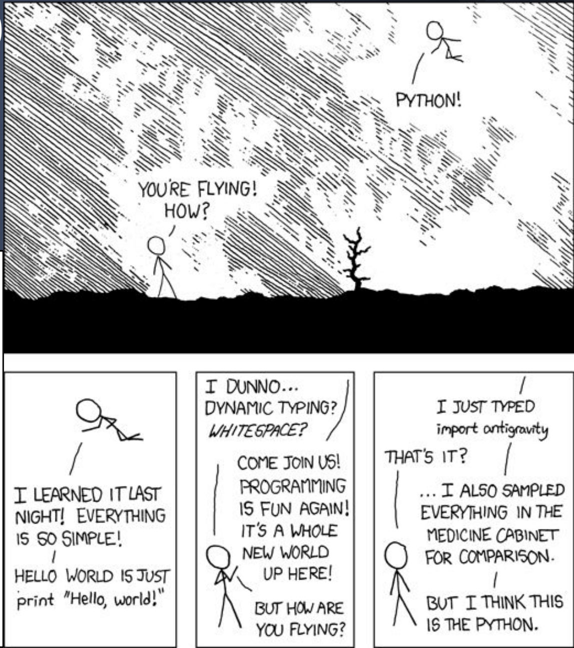
Side Quest Schedule (In the Thorn Pi Lab)

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
HAAAAAAAANDS 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

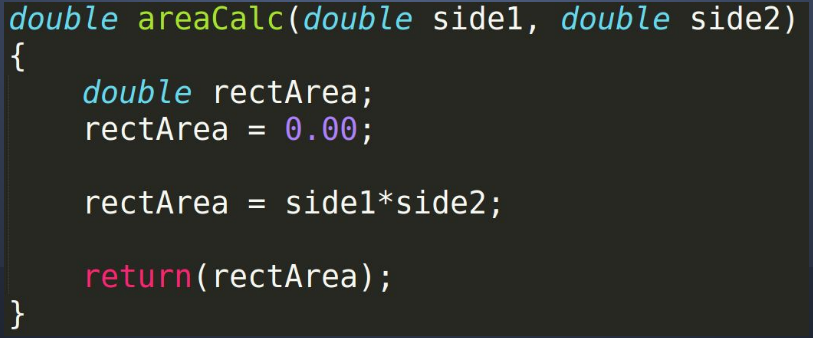
Function Definition

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

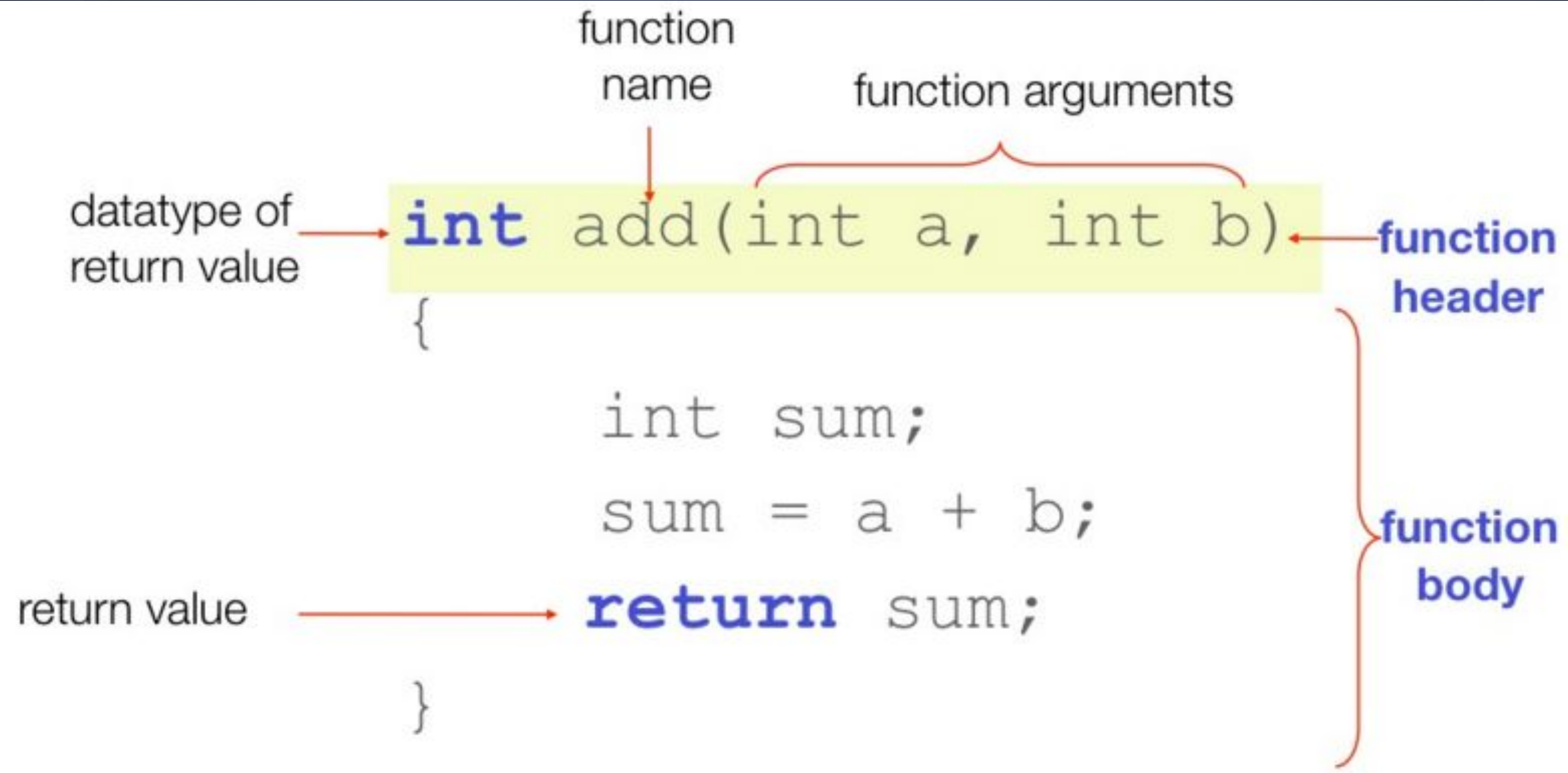
Input:Parameters (**double side1, double side2**)

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!

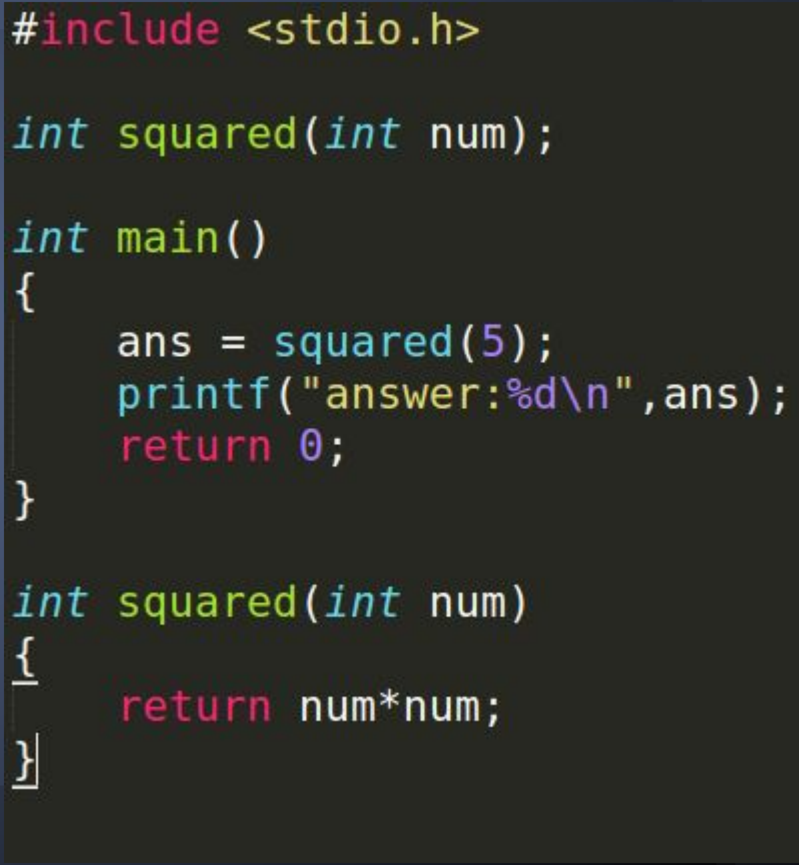
- put the whole function before main(how we’ve been doing it before)
- put a **function prototype** before main!

Function Prototype

includes:

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

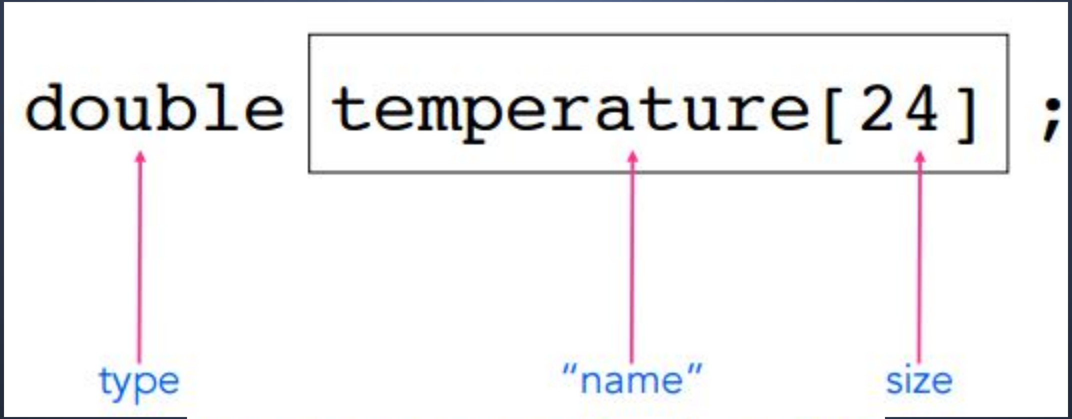
(basically the first line of our function def’n)



```
#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

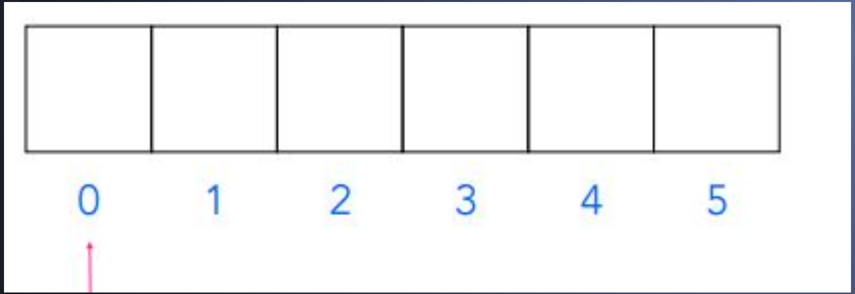
An array is a consecutive series of variables that share one variable name.



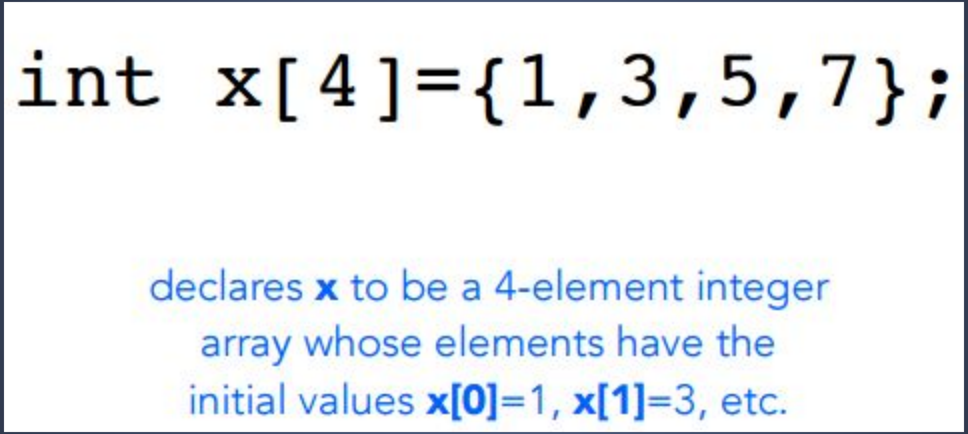
```
type name[size];
```

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!