

# Day 1: Introduction

THE UNIVERSITY OF  
**ALABAMA**<sup>®</sup>

# Agenda

- Syllabus/Schedule
- Tips for success
- Zybooks: Basic Shell
- Visual Studio Code demo of basic shell



Welcome. Today our goal is to go over the structure of the class along with tips for success. I will also introduce some of the tools used in this course (Blackboard, Zybooks, Visual Studio Code). I hope to have time to find out a bit about you as well

## Goals of course

- The structure of this class is designed to accomplish these three goals:
  - Learn and practice the skill of programming (in C for this class),
  - Apply the knowledge that you learn to the skill of programming,
  - Begin to apply processes that enable you to resolve and move past problems more quickly, and
  - Reflect on what it means to be successful at software creation (notice I said successful and not good...not sure what good means here and I doubt it matters).

THE UNIVERSITY OF  
**ALABAMA**<sup>®</sup>

## Class Design

- Complete zybooks sections (due dates visible on zybooks and the schedule)
- Quiz during first 5 minutes of class (will also serve as attendance)
- In class design and debugging activities
- Labs in recitation weekly (no labs this week, Labor Day week, and dead week)
- 6 projects due over course of semester
- Exams are tracing and programming

THE UNIVERSITY OF  
**ALABAMA**<sup>®</sup>

Syllabus/Blackboard

<https://ualearn.blackboard.com>



Go to blackboard and pull up syllabus

GO through each section

Discuss blackboard (official post grades, would submit assignment here but we use zybooks for submission)

A bit about you



THE UNIVERSITY OF  
**ALABAMA**<sup>®</sup>

## Tips for success

- Get good at finding bugs
  - Types
  - Copy errors
  - Misunderstandings about how a programming construct works
- Print out variables values every step of the way



Finally, let's take about debugging. The truth is that all software has bugs. It is the nature of the beast. The holy grail of writing software that runs perfectly after you type it in is not possible. In fact, I would say that it is not even important. Sometimes, we just type in the wrong thing (because we are human). Sometimes, we misunderstand how something works (because we are human). Sometimes, we have an idea of how something should be implemented and 80% through, we see that it is a bad idea (again because we are human). There is a certain amount of trial and error. I have been programming for 35 years. I often can't remember the dirty details of a language or an algorithm. I write a small part of the program and test it out to see if my assumptions were correct. If there were not, I have a small piece of code to change. I want you to start learning to develop iteratively. "Write a bit..check a bit...write a bit...check a bit." The truth is good software development is an iterative process. Treating it a such will lessen the time it takes to create something, greatly lessen your frustration and I daresay add the joy back into software creation.

## Tips for success

Exams are not for devising novel code. If you approach it that way, you will not finish in time and you will be frustrated. The exam is where you look through the catalog of program examples you have created and start with a program that is the closest to the exam question (which you have already made work).



The structure of this class is designed to accomplish these three goals:  
Learn and practice the skill of programming (in C for this class),  
Apply the knowledge that you learn to the skill of programming,  
Begin to apply processes that enable you to resolve and move past problems more quickly, and  
Reflect on what it means to be successful at software creation (notice I said successful and not good...not sure what good means here and I doubt it matters).

Learning to program is kind of strange. It is abstract. It is easily the most abstract thing you will ever encounter. It is so abstract that applying the knowledge requires that you use the knowledge. (I bet many of you used examples in Math to learn how to work a problem. This is similar) Therefore, in class, we will focus on skills where you apply the knowledge. If I use class time only for knowledge, you miss out on applying the knowledge in a manner where you can ask questions. You miss out on learning the processes of iterative development and write/check looping. You suffer in frustration alone, often spending too much time trying to unproductively fix an error.



When you finish this class, I want you to be confident in the skills you have learned and confident you are in the right major. Is this class easy? Nope. Playing an instrument is not easy. Making free throws is not easy. Whatever it is that Tiger does sure does not look easy. You will work. This class will be on your mind from the time you wake until you drift off to sleep. In the beginning, that is what it takes. We have a lot of material to cover, and we have designed this class to reward you for keeping up and for making the hard decisions between doing your out of class work and attending that football game after party.

The more you code, the more comfortable you will become with coding. Exams are not for devising novel code. If you approach it that way, you will not finish in time and you will be frustrated. The exam is where you look through the catalog of program examples you have created and select to modify the program that is the closest to the assignment.

You can think of me as your tour guide for this cruise. I am here to help you learn how to learn the skills.


# Zybooks: Basic Shell

PARTICIPATION ACTIVITY 1.4.1: Program execution begins with main, then proceeds one statement at a time.

Start ☐ 2x speed

```
#include <stdio.h>

int main(void) {
    int wage;
    wage = 20;
    printf("Salary is ");
    printf("%d", wage * 40 * 52);
    printf("\n");
    return 0;
}
```



Salary is 41600

Captions ^

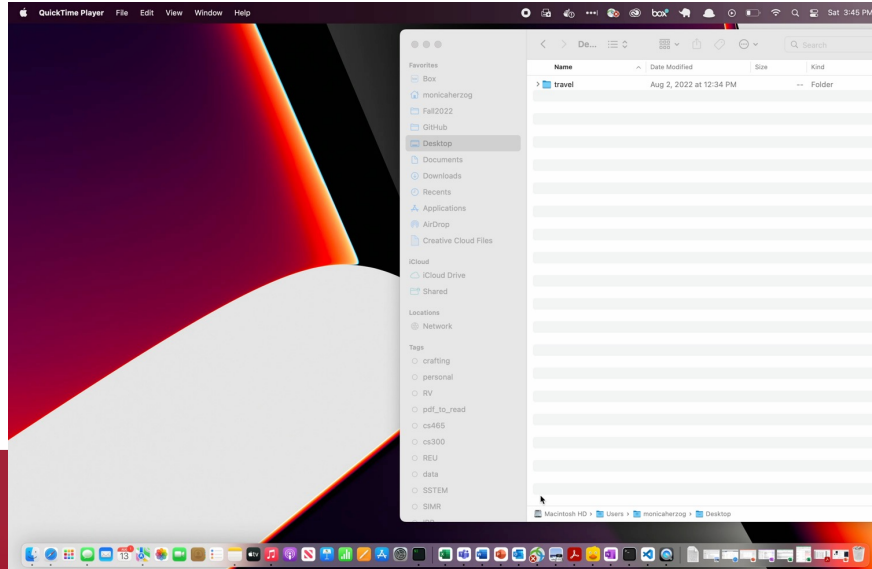
1. A program begins executing statements in main(). 'int wage' declares an integer variable.
2. The printf statement outputs 'Salary is ' to the screen at the cursor's present location.
3. This printf statement outputs the result of wage \* 40 \* 52, so 20 \* 40 \* 52 or 41600.
4. The printf statement with '\n' moves cursor to next line.
5. 'return 0' statement ends the program.

Feedback?

THE UNIVERSITY OF  
**ALABAMA**<sup>®</sup>

A program begins executing statements in main(). 'int wage' declares an integer variable. 'wage = 20' assigns wage with 20.  
The printf statement outputs 'Salary is ' to the screen at the cursor's present location.  
This printf statement outputs the result of wage \* 40 \* 52, so 20 \* 40 \* 52 or 41600.  
The printf statement with "\n" moves cursor to next line.  
'return 0' statement ends the program.

## Your first program: demo.c



Let's look at a programming visual studio code. We will create a basic shell like zybooks but even more basic.

Let's start visual studio code. It is a Microsoft program but it is cross platform. That means that it is available for use on windows and mac and students should have a fairly similar experience.

Because you will need your code examples, it makes sense to create a folder to keep them in. I put it on the desktop because it is easy to find. If you are using Windows Subsystem for Linux, your files will be organized a bit differently. Windows Cygwin (which reflect the instructions in the basics doc, should look similar to mac os.

I create the folder when I open the directory. Opening the directory just means that you will see all the files in that directory. I will execute file operations on the command line from inside Visual studio. I will also show the finder window so you can see how the command line and visual studio code operations are reflected on the file system.

I could create a file. You may have as many as 100 or more C programs before the end of the semester. You may want to further subdivide and organize the files. I

chose to use the day as the subdirectory

There are two icons on the folder. One is a piece of paper with a plus. That creates a new file. Our code goes in this file. The other is a file folder with a plus. That adds a new folder, which will help us with organization.

After creating a day 1 folder, I create a file called hello.c.

The most basic program that each student writes on the first day is called hello world. All it does is print hello world. This program illustrates what must be present in a working C program file.

Because no program executes without either reading or writing to/from the user, we have to include `stdio.h`. We will provide more insight as to why this is true later.

When we save the program, other applications can read the text we put inside.

The command line in Visual Studio Basic is reached by choosing terminal new terminal. We need to make sure we are in the correct place. The files are stored in a tree-like hierarchy. We move up and down the tree. You can imagine if everything was in one directory, it would be hard to find anything. The price of better organization is that we have to move around the tree to access the files we need.

The program is how we encode the solution to a problem. It is cryptic but is it not cryptic enough for the computer to understand. We will use a compiler to translate from the C program to an executable that we can run. I know I am using a lot of new terms. I will continue to use them so that you can add them to your professional vocabulary. However it is perfectly OK and expected for you to say "What is that again?"

@9:30 We will create a new program. We will start with hello C so we don't have to retype the basic shell we need. Getting good at reusing working code and modifying/adding to it to perform a new function is a good approach to coding. I usually grab a copy of something I have created previously or I will google to find an example. You cannot google during the exam so it is in your best interest to organize and document the code you create.

Programs are not useful unless they work on novel data. You wouldn't create the same paycheck each time and you wouldn't put the same passengers on the same plane. So we not only do computations, but we must create variables to work with novel data provided by the user. `appleCount` and `pearCount` are two variables meant to hold...well the count of apples and the number of pears. We have hard code

the values. We will more often retrieve values from files or the users or sometimes even the Internet. We will start calculations Friday and next week. For now we will just print them out. Your zybooks assignment will start with creating variables by using declarations and setting variable values, either when you declare the variable or by reading it in from the users.

## Friday

- Zybooks 1.1-1.6 due at 10am
- Quiz over Zybooks 1.1-1.6
- Install Visual Studio Code (see basics on blackboard)
- Office hours:
  - Tuesday 3-4 via zoom
  - Thursday 12-1pm SEC3435



I have posted this video. It is meant to bridge the gap between the zybooks programming facility and the visual studio code that you have on your computer. Submissions will be via zybooks. In some cases, you will have limited submissions so you will need the ability to program and test locally.

We have a lot to cover this semester so we are leveraging the excitement of a new semester by getting right to it. I have office hours tomorrow for anyone that needs help setting up their computer.

Friday, there will be a short quiz over the zybooks materials. The questions will use a similar format to the zybooks questions. You will need your turning mobile app loaded and a purchased key. We will complete a debugging exercise in class so have your computers charged and your compilers ready. You will work in pairs.