Variable Declaration and Data Types

COP 3223C – Introduction to Programming with C

Fall 2025

Yancy Vance Paredes, PhD

User-defined Identifiers

- In the real world, we use **names** to refer to *people or things*
- In programming, these names are called **identifiers**
- Identifiers are user-defined names for variables, functions, etc.
- But… not every name is allowed; certain rules must be followed

Valid User-defined Identifiers

- **Must consist** of only of letters, digits, and underscores
- **Cannot** begin with a digit
- **Cannot** be a C reserved word
- An identifier defined in a C standard library should not be redefined

Valid or Not?

```
num1
num_1
Num_1
Pneumonoultramicroscopicsilicovolcanoconiosis

_________
2gether
ag@in
int
```

Note: Identifiers are **case-sensitive**!

Notes

- A good framework to follow is to assume that an identifier is valid unless you can find at least one rule that it violated

- Think of it as innocent unless proven guilty!

Best Practices

- Use meaningful names (self-explanatory)

- Follow consistent naming convention:

  `camelCase`

  `snake_case`

  `PascalCase`

Reserved Keywords /1

- Words that have **special meaning** for the programming language

- IDEs often **change the colors** of these words to make it obvious

Some Sample Reserved Words in C		
auto	extern	signed
break	float	sizeof
case	for	static
char	if	struct
const	inline	switch
continue	int	typedef
default	long	union
do	register	unsigned
double	restrict	void
else	return	volatile
enum	short	while

Variable

- **Named location** in the memory that can **hold data** of a given *type*
- The data **(value) can change**
- Try visualizing the memory

Memory Location	Memory Content
#E	
#D	
#C	
#B	
#A	

stack space

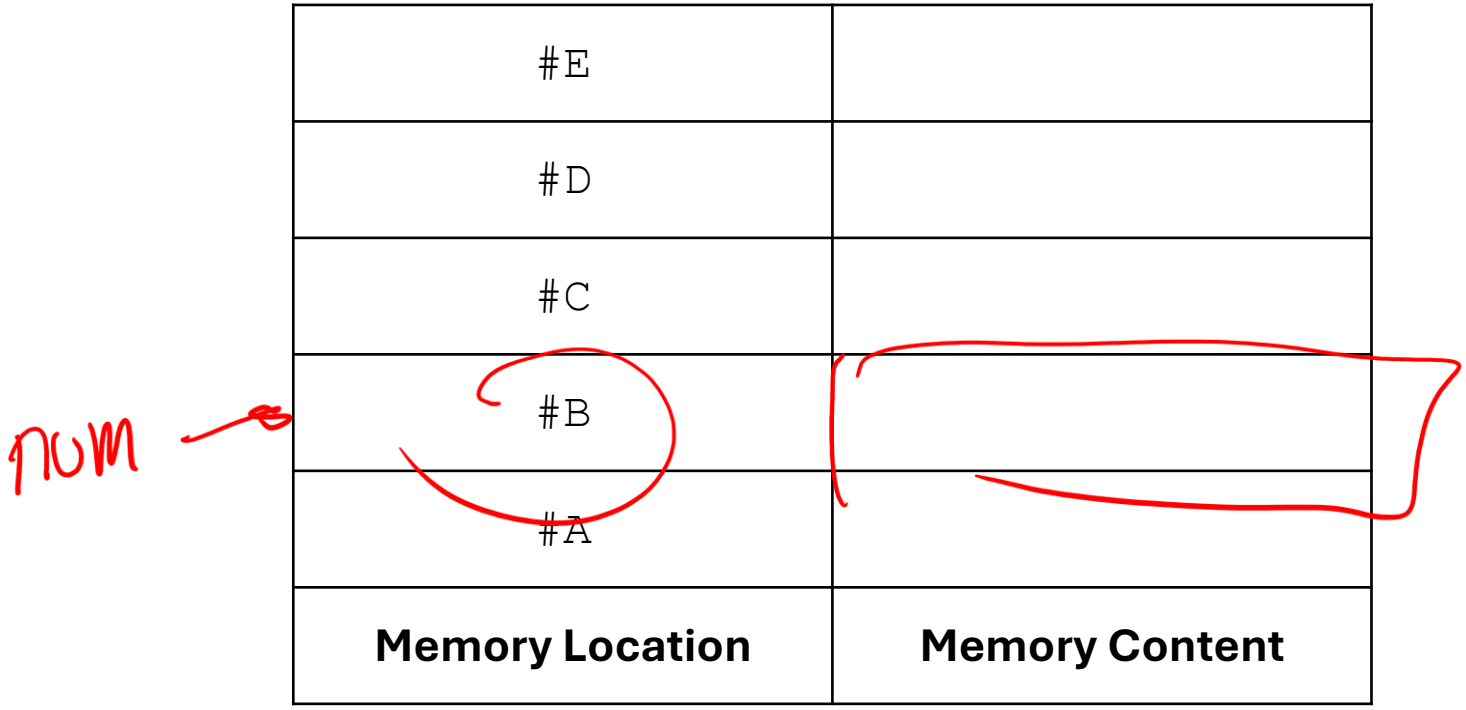
Variable Declaration /1

- Informs the compiler the **existence of a variable** and the **kind of data** it can store and **what operations you can perform** with it

- Syntax:

```
type identifier;
```

```
int num;
double average;
char choice;
```



Variable Declaration /2

- You can declare **multiple variables** in a single statement

- Syntax:

```
type identifier1, identifier2;


int num1, num2;
double average1, average2;
char choice1, choice2, choice3;
```

Important Notes /1

- If you’re coming from Python, the idea of **declaring variables** before using them may be new

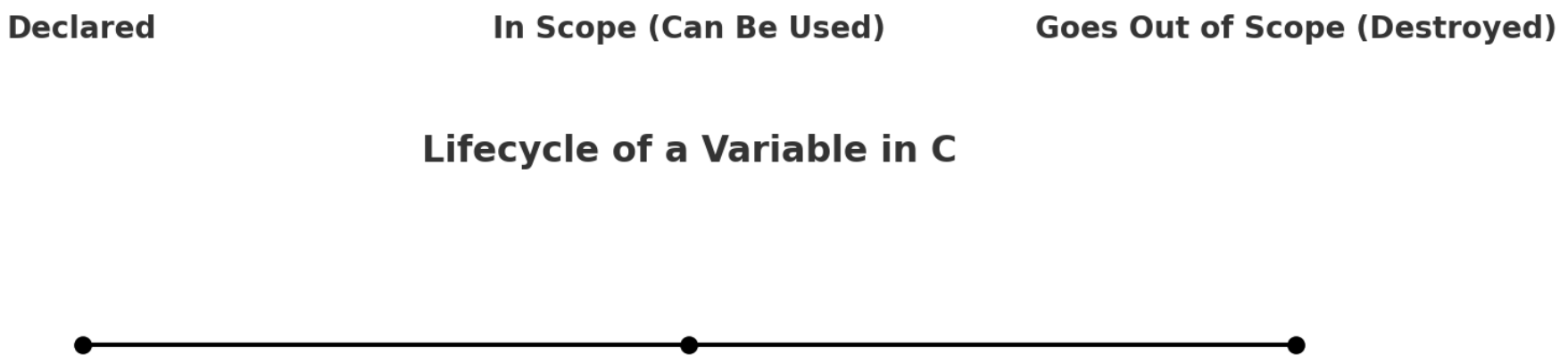
- Variables must be declared with a type **before they can be used**

Important Notes /2

- A **variable's lifetime** begins at the point of its declaration
- The **order of lines matters**; you cannot use a variable before it has been declared
- For now, we will declare all our variables at the beginning of the **`main()`** function

Variable’s Lifecycle

Begins when it is **declared**, continues while it is in **scope**, and ends when the block or function it belongs to finishes execution (i.e., the variable becomes ***out of scope***)



Notes

A variable's **scope** refers to the region of a program where a variable is visible (can be accessed) and alive (exists in the memory)

Practice

Walk through the code one statement at a time. When a variable is declared, add it to your diagram of the main memory. When does a variable come into existence? When is it destroyed?

```c
#include <stdio.h>

int main(void) {
    int num;
    int num2;
    double value;


    return 0;
}
```

Notes

To be more precise, the type of variable being referred to earlier is a **local variable**

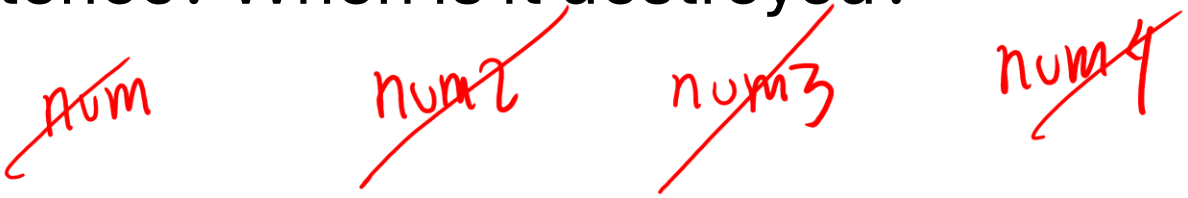
Local variables are variables that are declared inside a function or a block, such as **{ }**

Don’t forget when a variable’s lifecycle ends

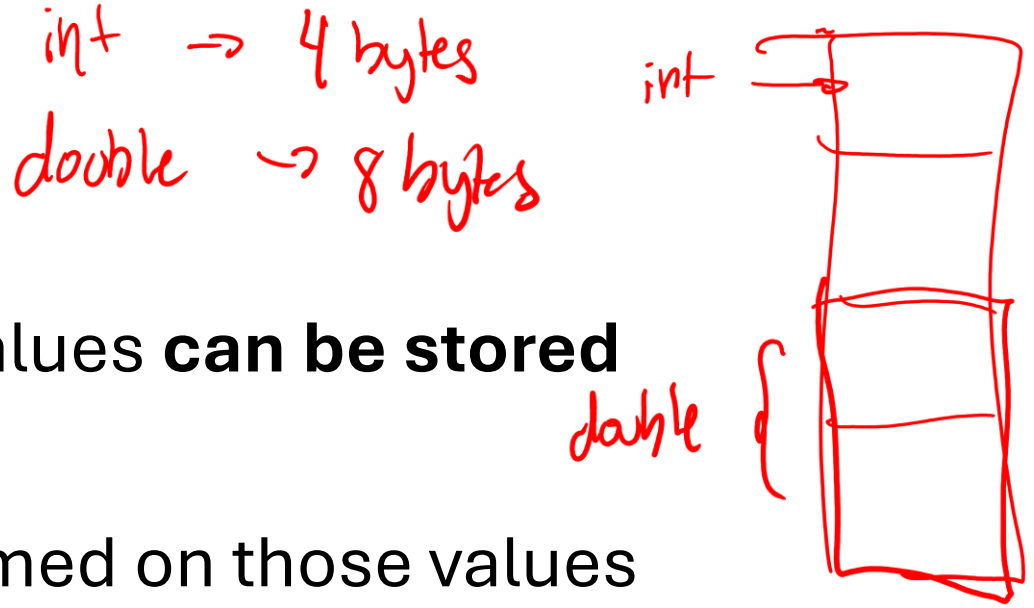
Practice

Walk through the code one statement at a time. When a variable is declared, add it to your diagram of the main memory. When does a variable come into existence? When is it destroyed?

```c
#include <stdio.h>

int main(void) {
    int num;

    // notice the { }
    {
        int num2;
        int num3;

    }

    int num4;


    return 0;
}
```



Data Types /1



- Specify what kind of data and values **can be stored**
- What **operations** can be performed on those values
- Determines **how much space** needs to be reserved (allocated)

Data Types /2

Fundamental data types

`int`, **`double`**, **`float`**, and **`char`**

Type	Description
`int`	integer (whole numbers)
`double`	number with fractional part (real numbers)
`float`	similar to double but smaller memory requirement
`char`	a single character (ASCII)

Data Types /3

Type	Range in Typical Implementation*
int	-2,147,483,647 … 2,147,483,647
double	10^{-307}…10^{308} (15 significant digits)
float	10^{-37}…10^{38} (6 significant digits)

long

unsigned

**This is a range which means it varies from system to system. You don't need to memorize exact size, but just understand there are limitations!*

Your Turn!

Do a research on the keyword **`unsigned`** and determine how to declare a variable using this keyword.

Data Types /4

- Note that `double` and `float` **are not interchangeable**
- `double` has twice the **precision** of `float`
- You may lose information if you use them interchangeably
- More on this in future discussions

Data Types /5

A = 65 a = 97
B b = 98

- The `char` represents an individual character value (ASCII value)

  Letter, digit, or special symbol

A a E

- **Case-sensitive** because the **ASCII code** of **A** is different from **a**
- The value is enclosed with **' (single quote)**

character 'A'

- **Examples:** `'A'`, `'a'`, `'8'`, `'?'`, `'\n'`

'8' ≠ 8

String "Hello World"

"A"

Escape Sequence Characters

Escape Sequence	Meaning
\a	Alert
\b	Backspace
\n	Newline
\t	Horizontal Tab
\v	Vertical Tab
\\	Backslash
\'	Single Quote
\"	Double Quote
\?	Question Mark
%%	Percent Symbol

'\n'

\t

printf(" " n)

Your Turn!

Write a program that outputs exactly the following text (including the double quotes)

```
"Hello World"
```

Constant /1

- A variable that is declared and assigned a value that **cannot be modified** during execution of a program

- Uses the reserved word **`const`**

- **Example:** `const double PI = 3.1416;`

- **Note:** *Different from a macro constant*

Constant /2

The following is another way of defining a constant

```
#define PI 3.1416
```

This is a preprocessor directive that you can think as a find and replace behavior (i.e., textual substitution)

Notice the absence of the semicolon. What happens if we add it?

Notes /1

An advantage of using the `const` allows the compiler to enforce type checking and scope

Notes /2

It is considered good practice to type the names of constants in all uppercase letters

```
NUM_COUNT
```

```
MAX_SIZE
```

Final Notes

- You can declare a variable only once in a given scope. Redeclaring a variable in the same scope leads to a syntax error. We will revisit this later once we discuss the concept of **variable scopes**.

- Think of variable declaration as the *birth* of a variable. Once born, you refer to it by its name.

- For now, declare all your variables at the start of `main()`.

Questions?