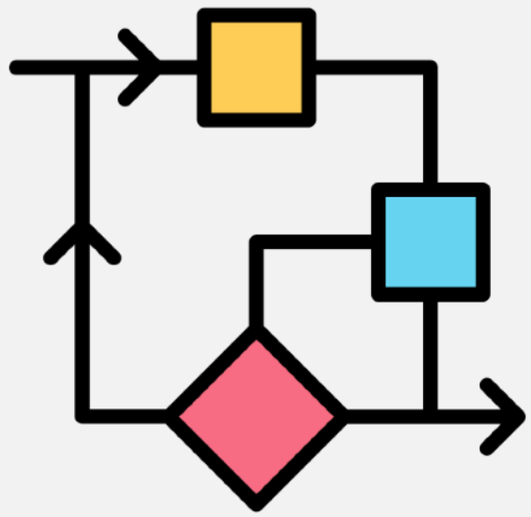
COP 3223H: Introduction to C Programming

Fall 2023

University of Central Florida

Dr. Kevin Moran

Week 4 - Class 1: Control Structures & Conditionals



Administrivia

- Small Programming Assignment 2 will come out later this week (Thurs/Fri)
  - I will be adjusting the timing of Small Programming Assignment 3 - moving to after Large programming assignment 1
- Quiz 5 will be due by Sunday at 11:59 pm
  - Released today
- Heads up on Exam 1
  - Will be next Friday (September 22nd)
  - We will review extensively in class

Today's Agenda

1. Review the User Defined Functions Concepts
2. Introduce Control Structures and Conditionals

Review User Defined Functions

Recap of Last Class - User Defined Functions

- Programmers can define their ***user defined functions*** to perform certain tasks
- Reasons
  - Code Reusability
  - More organized
  - Good Practice!
- ***Function Prototypes*** determine if the function will return anything and determine the amount of arguments.

Control Structures & Conditionals

Recap of Last Class (cont.)

- Control structures are a combination of individual instructions into a single logical unit with one entry point and one exit point
- Compound Statement is a group of statements bracketed `{ and }` that are executed sequentially.

```c
int main(void)
{

    printf("Hello World \n");
    return 0;
}
```

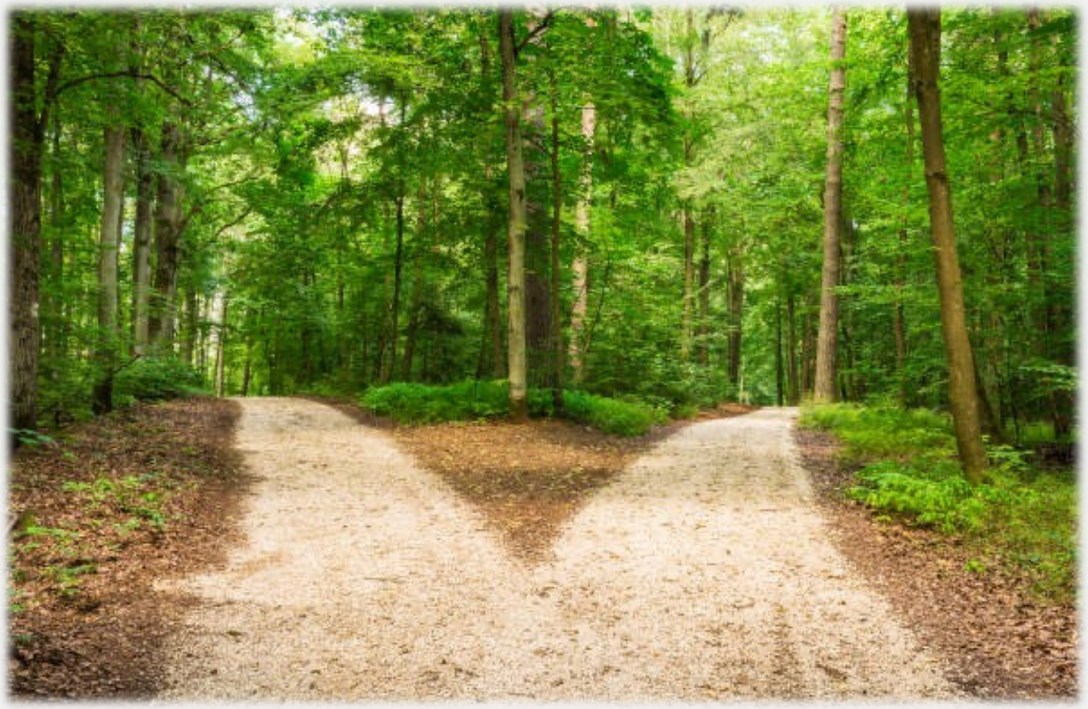
```c
int main(void)
{
    return 0;
}
```

Variable Scope

- Now that we have learned control structures, it is time to discuss variable scope.
- Scope is the level of access a variable has in a program run
- There are two types of scopes with variables.
    - Global Scope (*Bad!!!!!*)
    - Local Score (*Good!!!*)
- Global means all components (functions have access to the value and can manipulate it)
    - Why is that bad?
    - Never use Global Variables in this course unless Dr. Moran says it is ok
- Local means only the component within the control structure has access the value and can perform certain operations on it.
    - *Good Practice!!!*

What are Conditions

- Everyday we are always making decisions based on our surrounding environment.
  - Example 1: ***If*** it is cold outside, I will wear a jacket. ***If not***, I will wear a short sleeve shirt.
  - Example 2: ***If*** gas tank is close to "E", I will drive to Gas Station, ***If not***, I will continue to my destination.
- Conditions are expressions that has an outcome of either 0 or 1.
- C does not recognize Boolean types.
- Based on the condition outcome an action is executed.



Relational & Equality Operators

- When evaluating expressions, we make comparisons.
- There are 6 relational/equality operators.
  - Less than (<)
  - Greater than (>)
  - Less than or equal to (<=)
  - Greater than or equal to (>=)
  - Equal to (==)
  - Not Equal to (!=)
- Important! = and == are two different operators!!
  - = is the assignment operator
  - == is the equality operator

Relational & Equality Operators in C

Operator	Meaning	Type
<	less than	relational
>	greater than	relational
<=	less than or equal to	relational
>=	greater than or equal to	relational
(==)	equal to	equality
!=	not equal to	equality

Logical Tables

Operator	Example	English Meaning
<	x < 0	x is less than 0
>	power > MAX_POW	power is greater than MAX_POW
<=	x <= y	x is greater than or equal to y
>=	item >= Min_ITEM	item is less than or equal to Min_ITEM
(==)	mom_or_dad == ‘M’	mom_or_dad is equal to ‘M’
!=	num != SENTINEL	num is not equal to SENTINEL

Logical Operators

- An expression that uses one or more of the three logical operators
  - && (and)
  - || (or)
  - ! (not)
  - **&&** and || operators allows us to combine a set of conditions

- Examples:
  - `in_range = (num >= -10 && num <= 10)`
  - `is_letter = (letter == ‘a’ || letter == ‘b’)`
- ! operator complements (opposite result) the condition
- Examples:
  - `num1 == num2`
  - `!(num1 == num2)`

Logical Tables

The && Operator

Operand 1	Operand 2	Operand 1 && Operand 2
nonzero (T)	nonzero (T)	1 (T)
nonzero (T)	0 (F)	0 (F)
0 (F)	nonzero (T)	0 (F)
0 (F)	0 (F)	0 (F)

The ! Operator

Operand 1	! Operand 1
nonzero (T)	0 (F)
0 (F)	1 (T)

The || Operator

Operand 1	Operand 2	Operand 1 && Operand 2
nonzero (T)	nonzero (T)	1 (T)
nonzero (T)	0 (F)	1 (T)
0 (F)	nonzero (T)	1 (T)
0 (F)	0 (F)	0 (F)

Operator Precedence in C

Operator	Precedence
`function calls`	Highest
`! + - & (unary)`	
`* / %`	
`+ -`	
`< <= >= >`	
`!= ==`	
`&&`	
`||`	
`(=)`	Lowest

Acknowledgements

Slides adapted from Dr. Andrew Steinberg's COP 3223H course