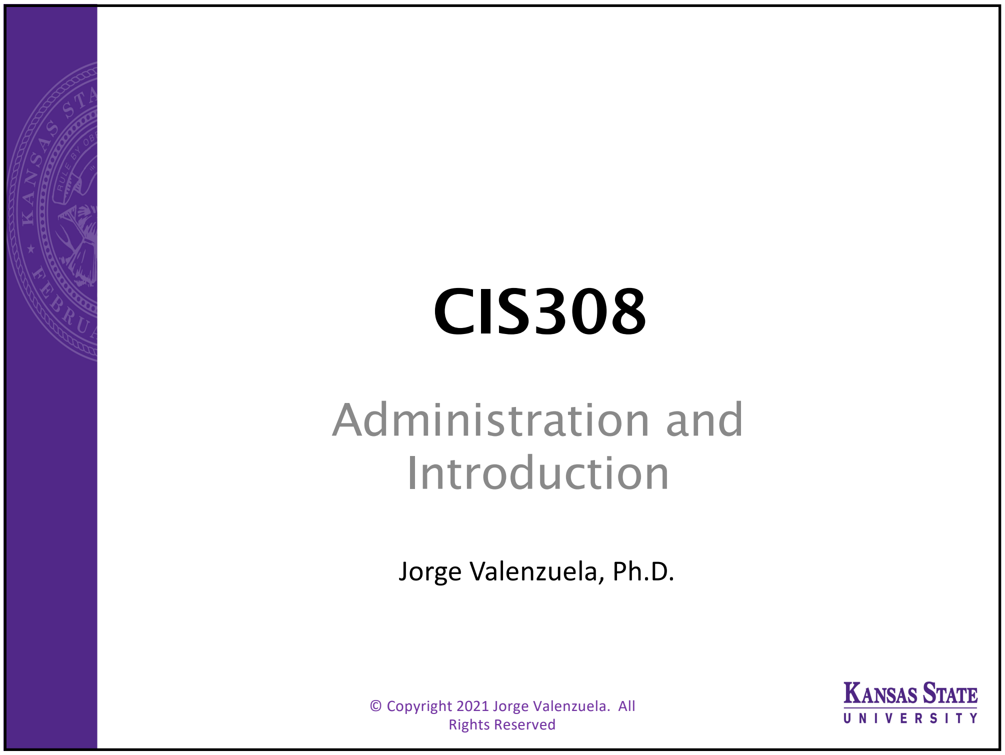
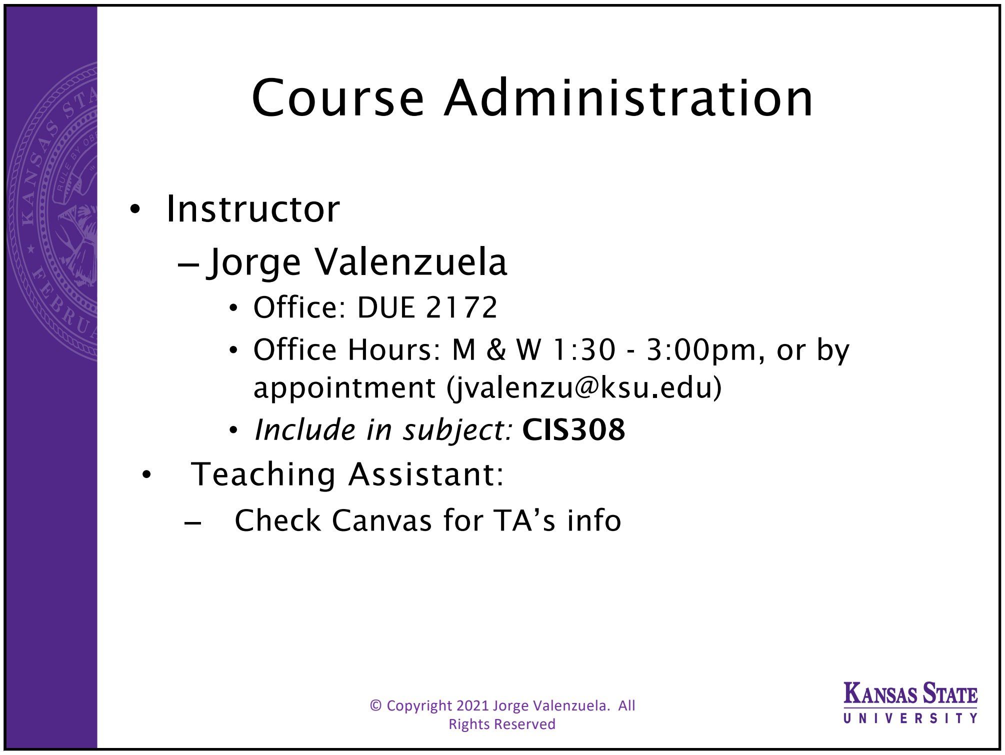


CIS308

Administration and Introduction

Jorge Valenzuela, Ph.D.

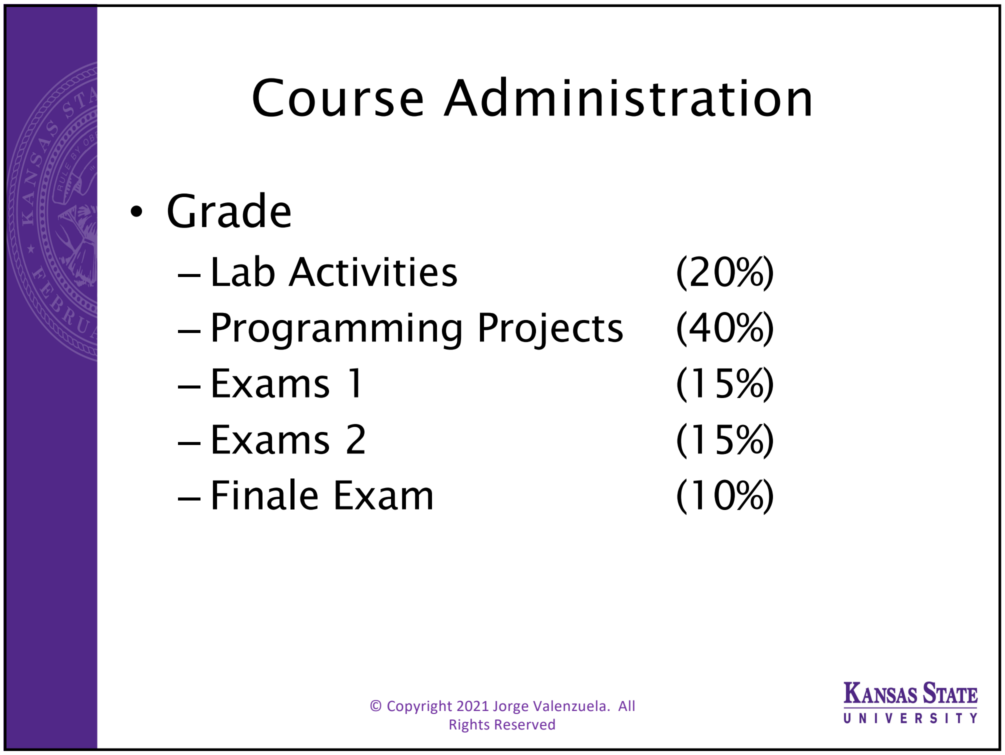
1



Course Administration

- Instructor
  - Jorge Valenzuela
    - Office: DUE 2172
    - Office Hours: M & W 1:30 - 3:00pm, or by appointment (jvalenzu@ksu.edu)
    - *Include in subject:* **CIS308**
- Teaching Assistant:
  - Check Canvas for TA’s info

2



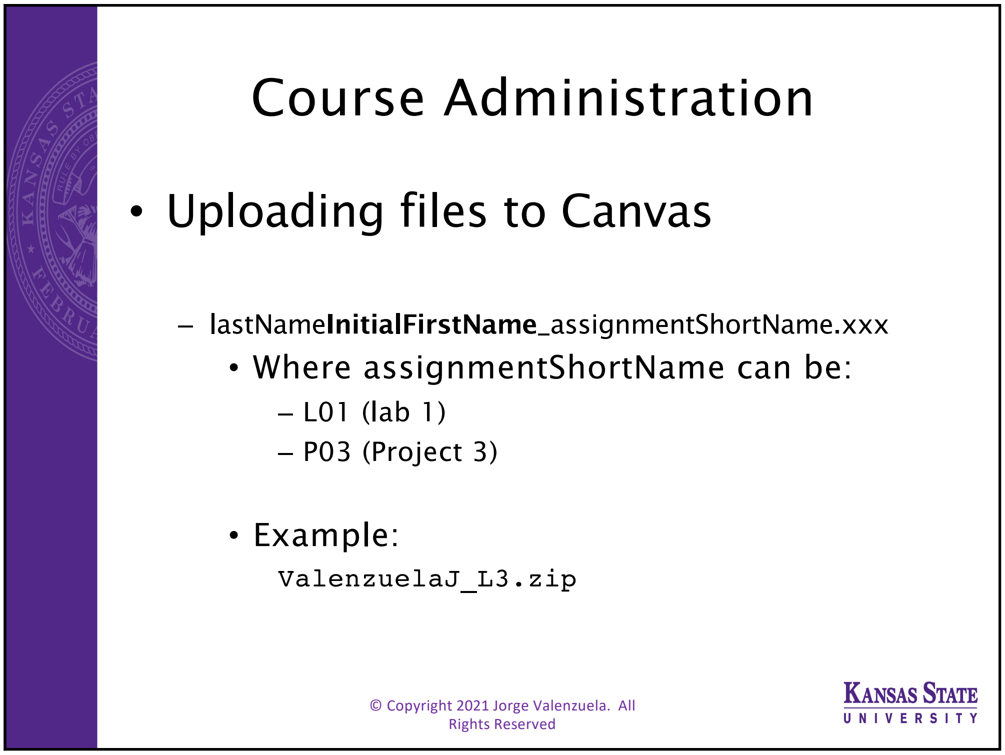
Course Administration

- Grade

– Lab Activities	(20%)
– Programming Projects	(40%)
– Exams 1	(15%)
– Exams 2	(15%)
– Finale Exam	(10%)

KANSAS STATE UNIVERSITY

3



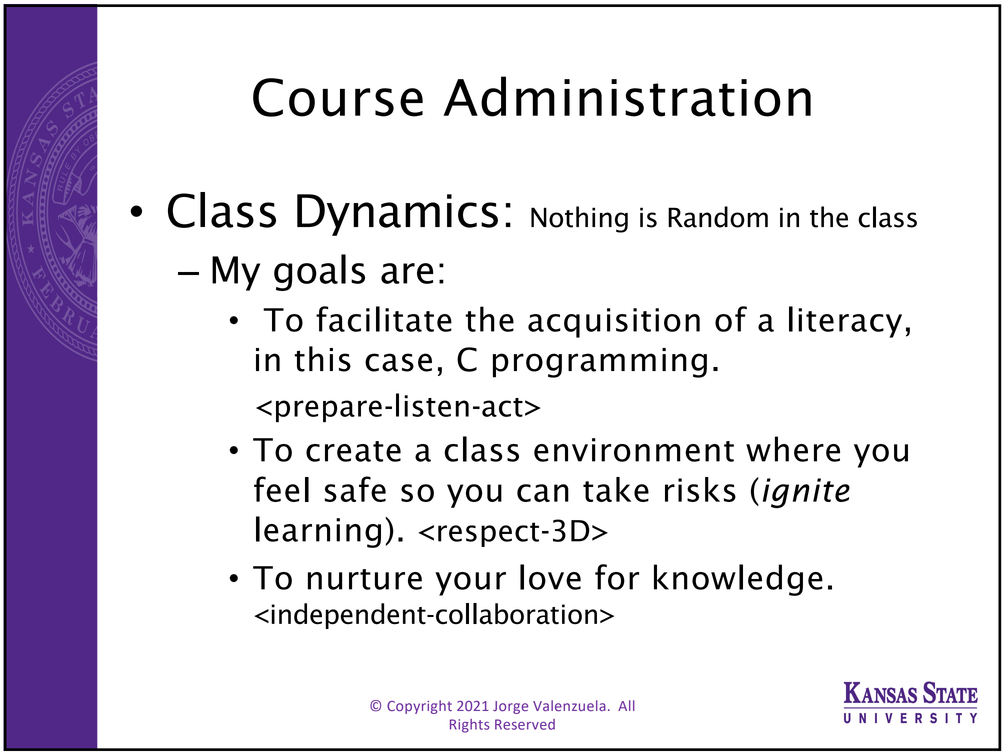
Course Administration

- Uploading files to Canvas
  - lastName**InitialFirstName**_assignmentShortName.xxx
    - Where assignmentShortName can be:
      - L01 (lab 1)
      - P03 (Project 3)
    - Example:

```
ValenzuelaJ_L3.zip
```

KANSAS STATE UNIVERSITY

4



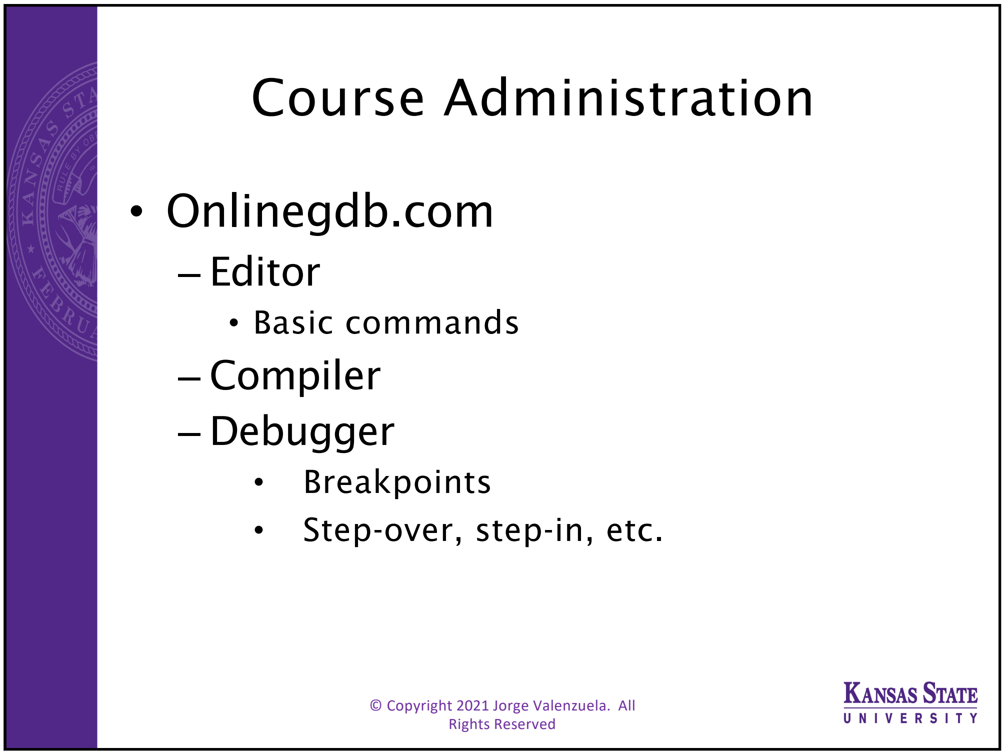
Course Administration

- Class Dynamics: Nothing is Random in the class
  - My goals are:
    - To facilitate the acquisition of a literacy, in this case, C programming.

      <prepare-listen-act>
    - To create a class environment where you feel safe so you can take risks (*ignite* learning). <respect-3D>
    - To nurture your love for knowledge.

      <independent-collaboration>

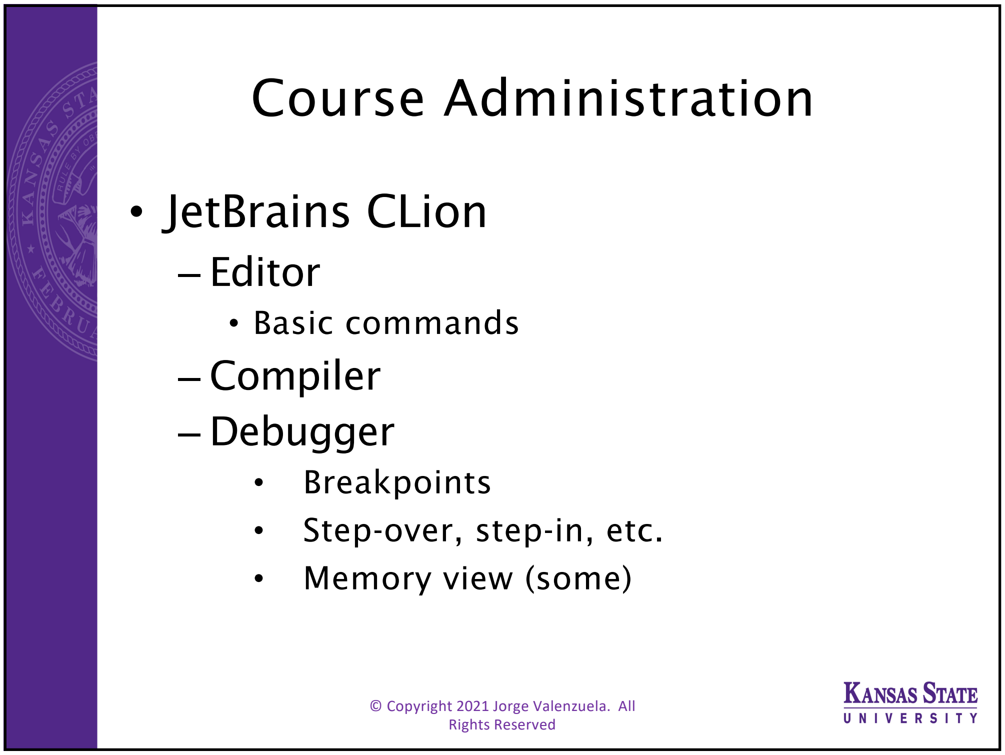
5



Course Administration

- Onlinegdb.com
  - Editor
    - Basic commands
  - Compiler
  - Debugger
    - Breakpoints
    - Step-over, step-in, etc.

6

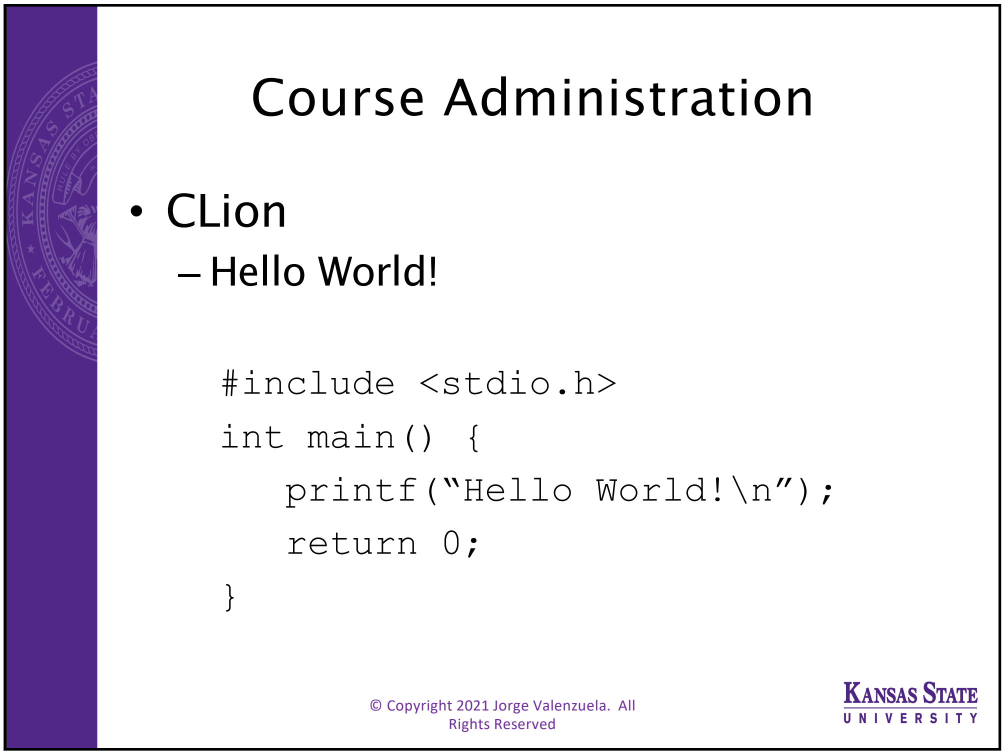


Course Administration

- JetBrains CLion
  - Editor
    - Basic commands
  - Compiler
  - Debugger
    - Breakpoints
    - Step-over, step-in, etc.
    - Memory view (some)

KANSAS STATE UNIVERSITY

7



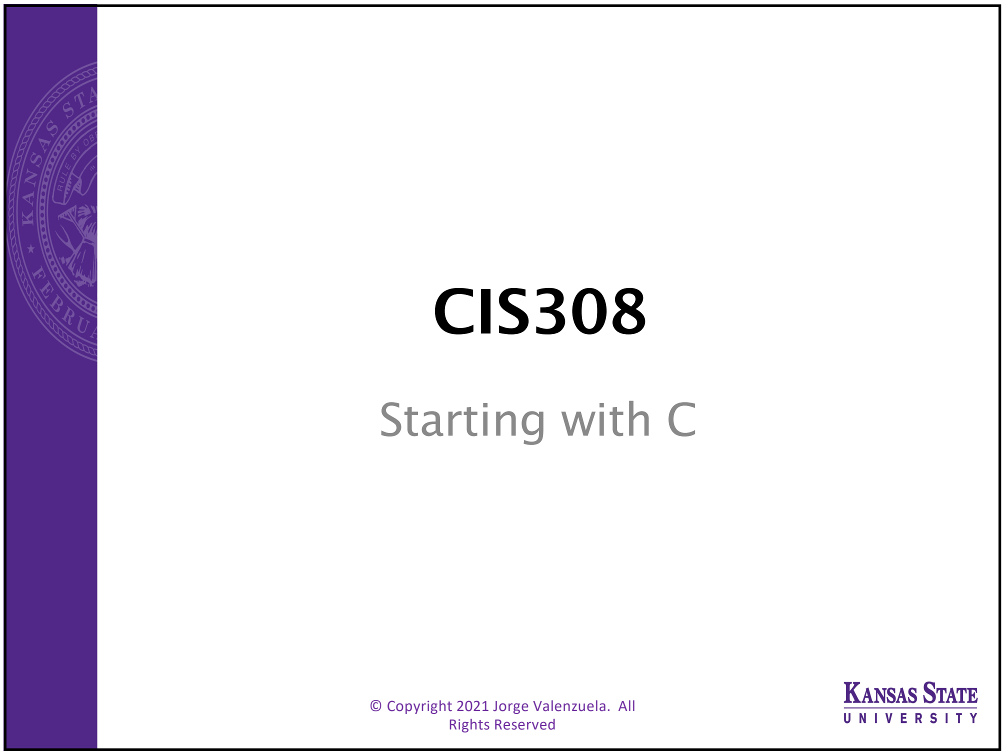
Course Administration

- CLion
  - Hello World!

```
#include <stdio.h>
int main() {
   printf(“Hello World!\n”);
   return 0;
}
```

KANSAS STATE UNIVERSITY

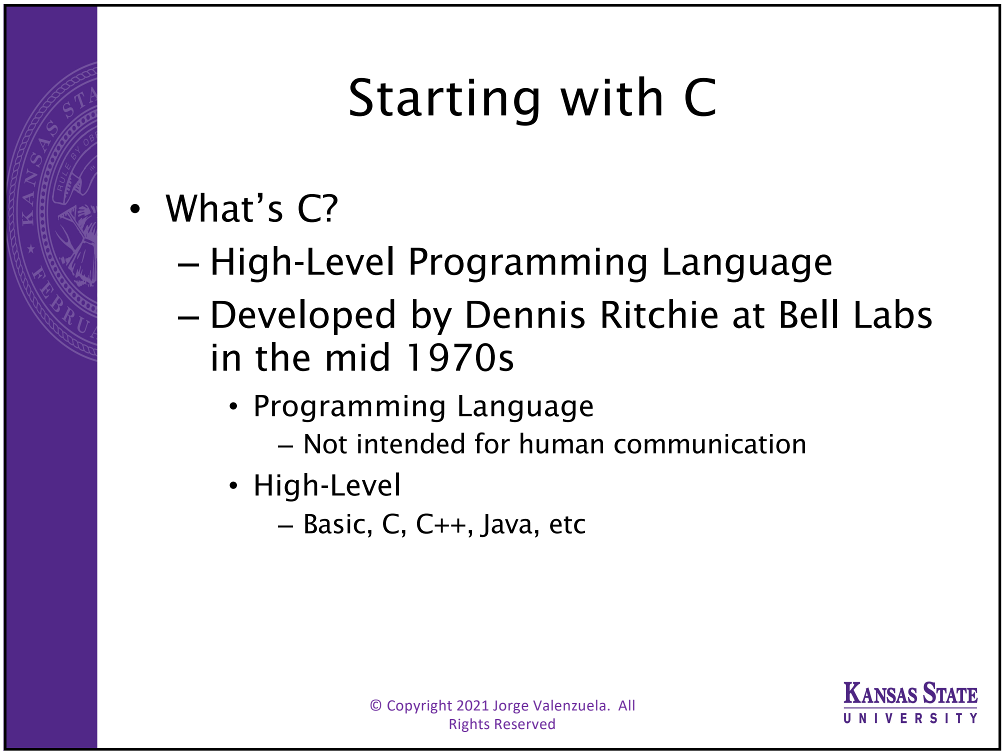
8



CIS308

Starting with C

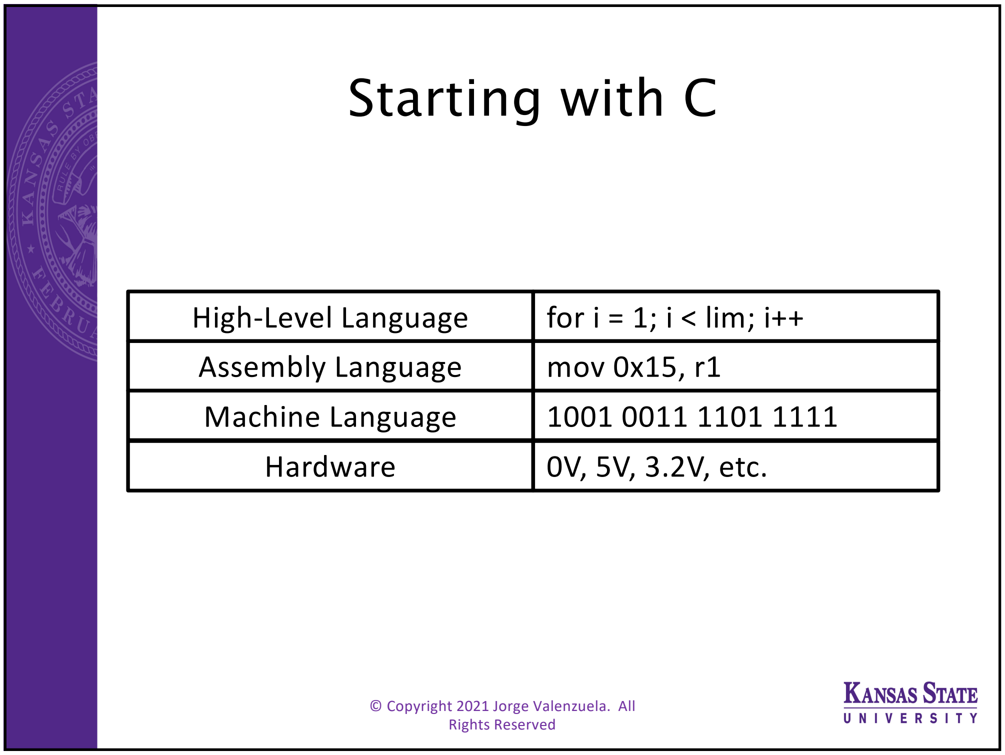
9



Starting with C

- What's C?
  - High-Level Programming Language
  - Developed by Dennis Ritchie at Bell Labs in the mid 1970s
    - Programming Language
      - Not intended for human communication
    - High-Level
      - Basic, C, C++, Java, etc

10

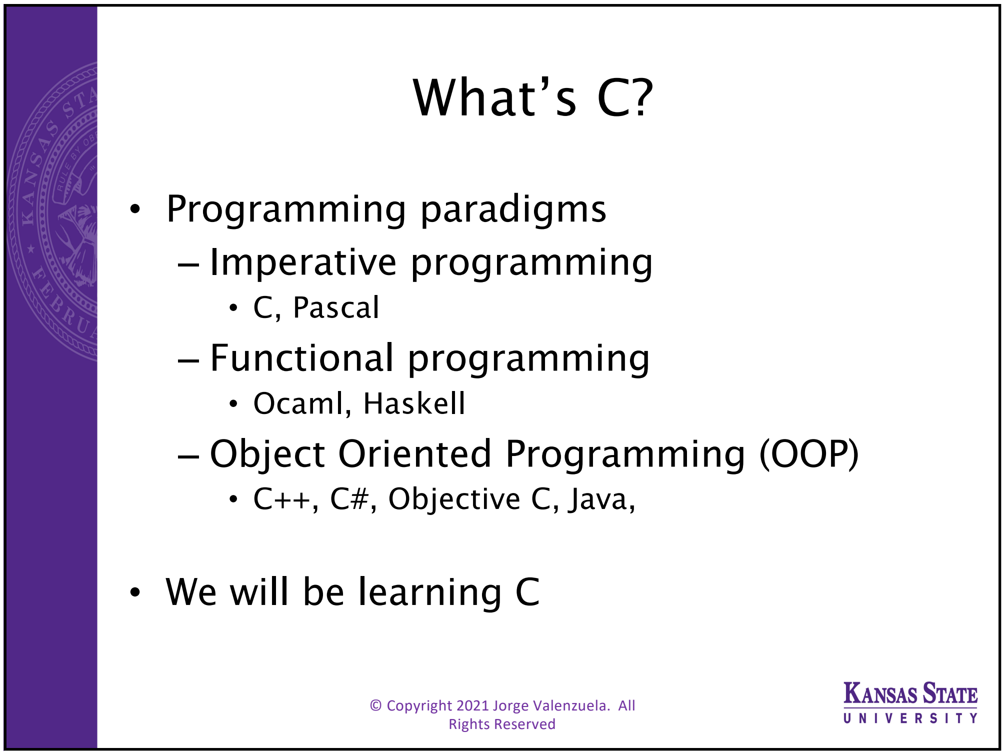


Starting with C

High-Level Language	for i = 1; i < lim; i++
Assembly Language	mov 0x15, r1
Machine Language	1001 0011 1101 1111
Hardware	0V, 5V, 3.2V, etc.

KANSAS STATE
UNIVERSITY

11

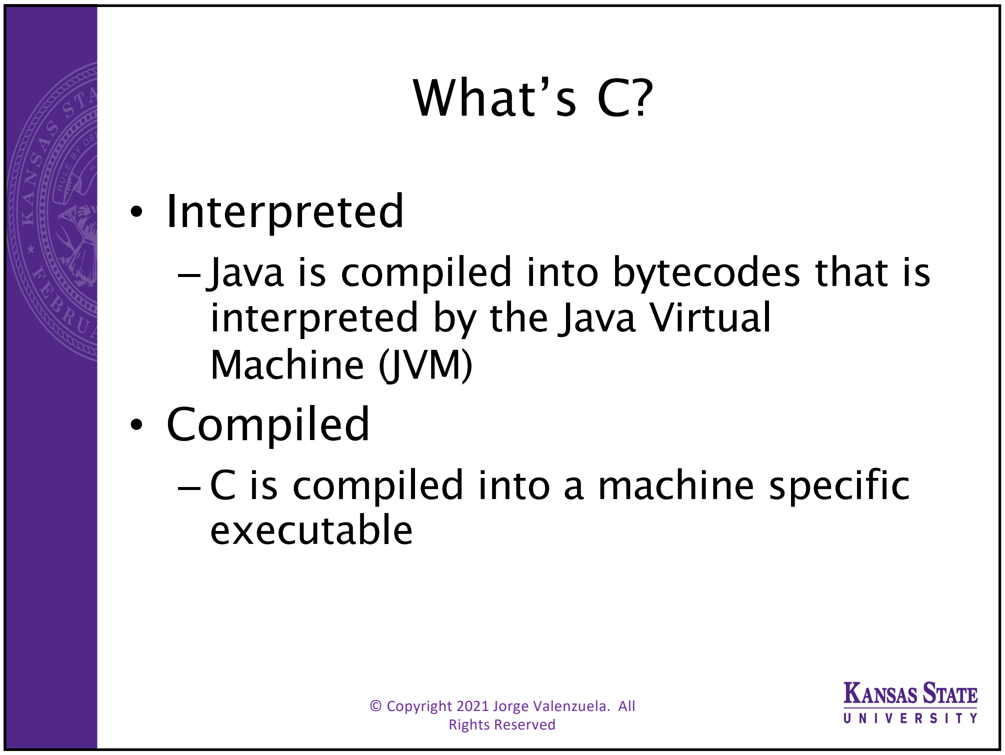


What’s C?

- Programming paradigms
  - Imperative programming
    - C, Pascal
  - Functional programming
    - Ocaml, Haskell
  - Object Oriented Programming (OOP)
    - C++, C#, Objective C, Java,
- We will be learning C

KANSAS STATE
UNIVERSITY

12

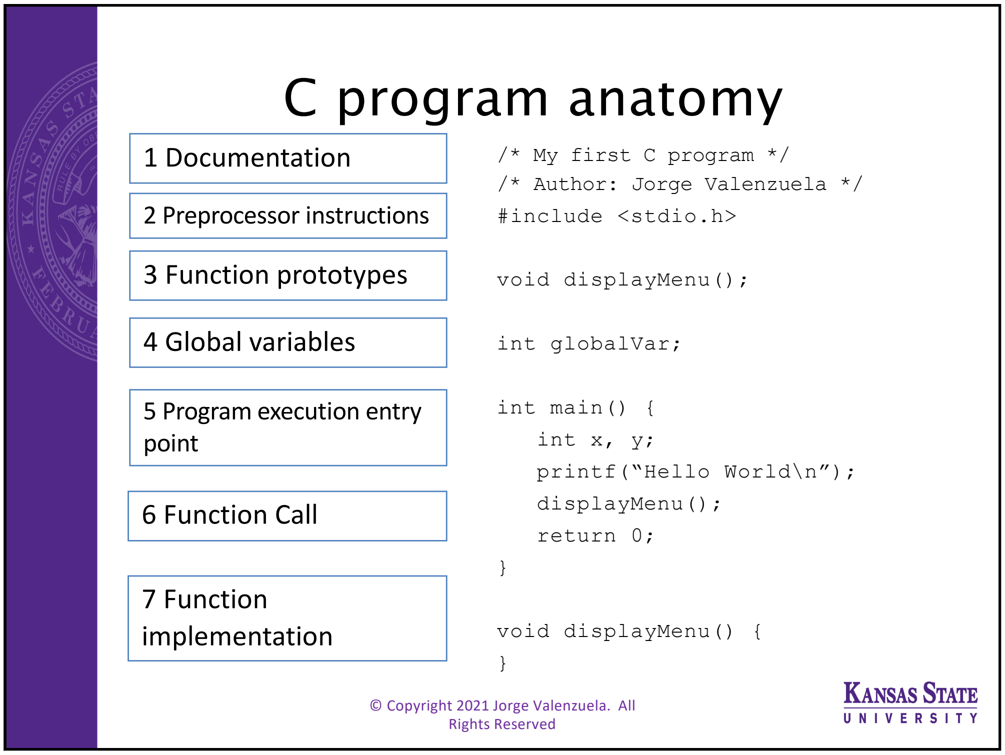


What’s C?

- Interpreted
  - Java is compiled into bytecodes that is interpreted by the Java Virtual Machine (JVM)
- Compiled
  - C is compiled into a machine specific executable

KANSAS STATE UNIVERSITY

13



C program anatomy

1 Documentation

2 Preprocessor instructions

3 Function prototypes

4 Global variables

5 Program execution entry point

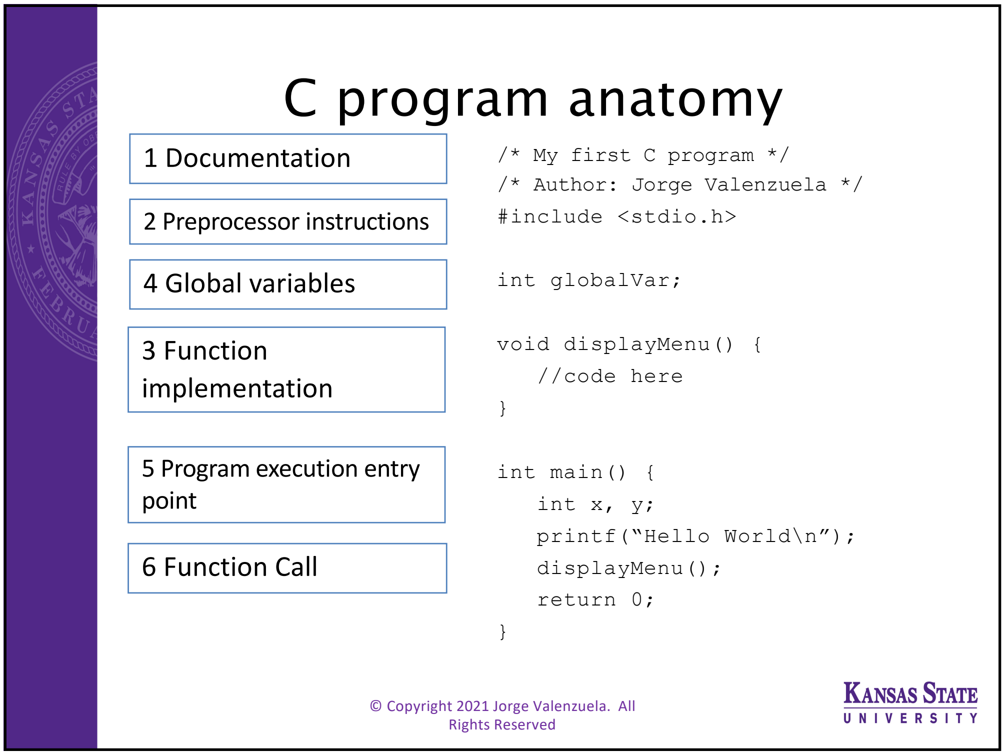
6 Function Call

7 Function implementation

```
/* My first C program */
/* Author: Jorge Valenzuela */
#include <stdio.h>

void displayMenu();

int globalVar;

int main() {
   int x, y;
   printf(“Hello World\n”);
   displayMenu();
   return 0;
}

void displayMenu() {
}
```

KANSAS STATE UNIVERSITY

14



C program anatomy

1 Documentation

2 Preprocessor instructions

4 Global variables

3 Function implementation

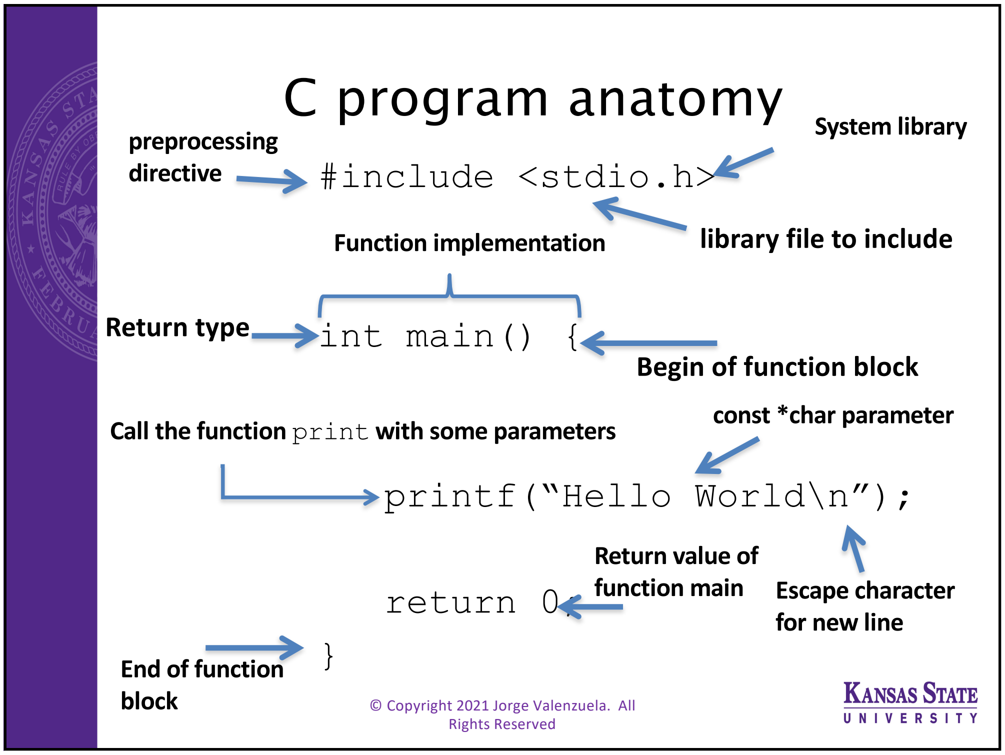
5 Program execution entry point

6 Function Call

```
/* My first C program */
/* Author: Jorge Valenzuela */
#include <stdio.h>

int globalVar;

void displayMenu() {
   //code here
}

int main() {
   int x, y;
   printf(“Hello World\n”);
   displayMenu();
   return 0;
}
```

KANSAS STATE UNIVERSITY

15

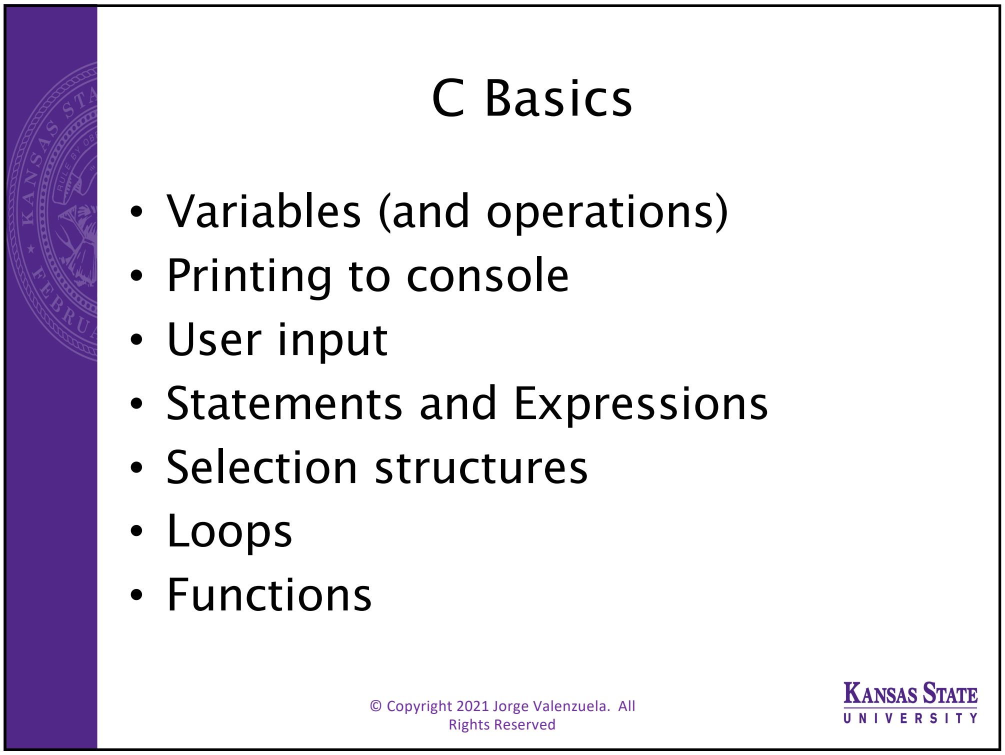


C program anatomy

```
#include <stdio.h>

int main() {

    printf(“Hello World\n”);

    return 0;

}
```

KANSAS STATE UNIVERSITY

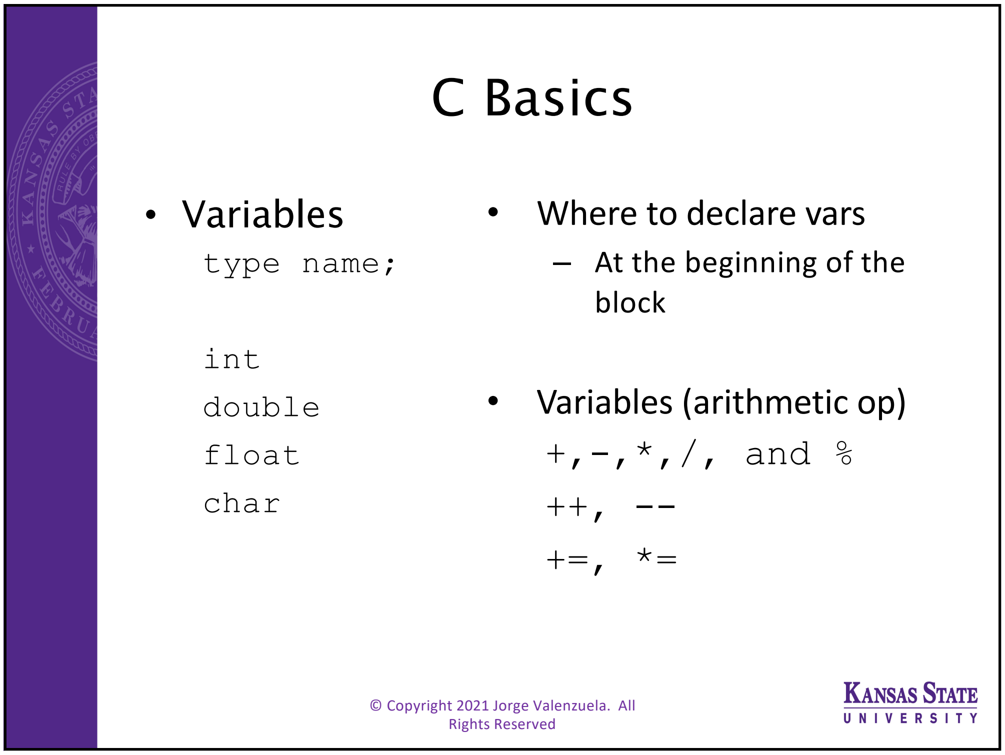
16



C Basics

- Variables (and operations)
- Printing to console
- User input
- Statements and Expressions
- Selection structures
- Loops
- Functions

17



C Basics

- Variables

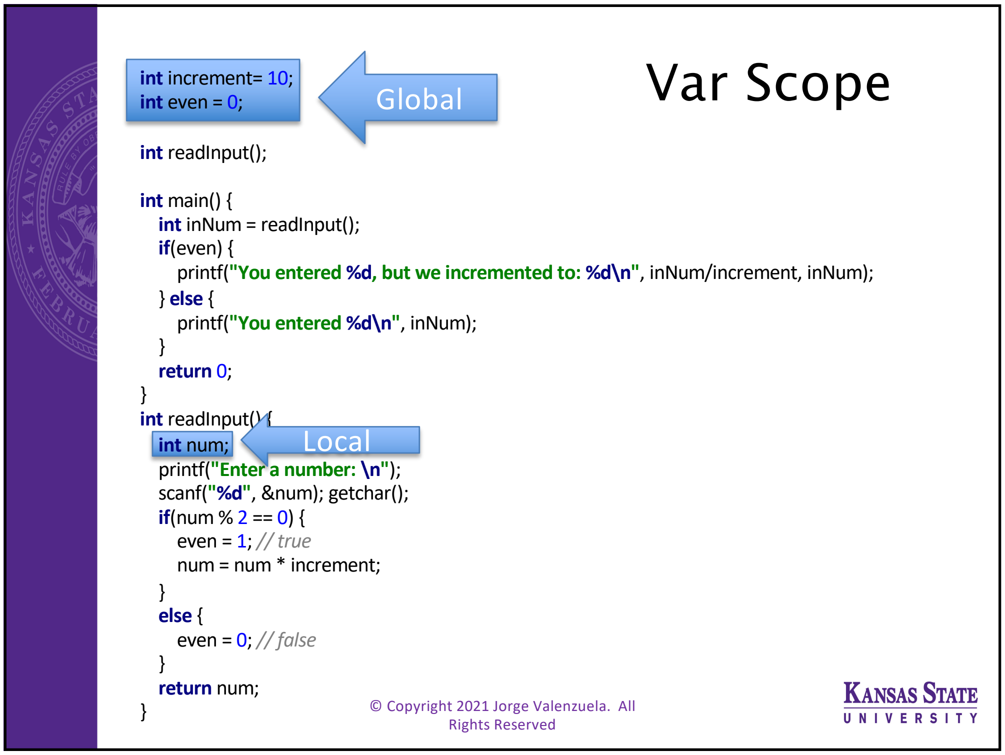
```
type name;

int
double
float
char
```

- Where to declare vars
  - At the beginning of the block

- Variables (arithmetic op)

```
+,-,*,/, and %
++, --
+=, *=
```

18



Var Scope

```
int increment= 10;
int even = 0;
```

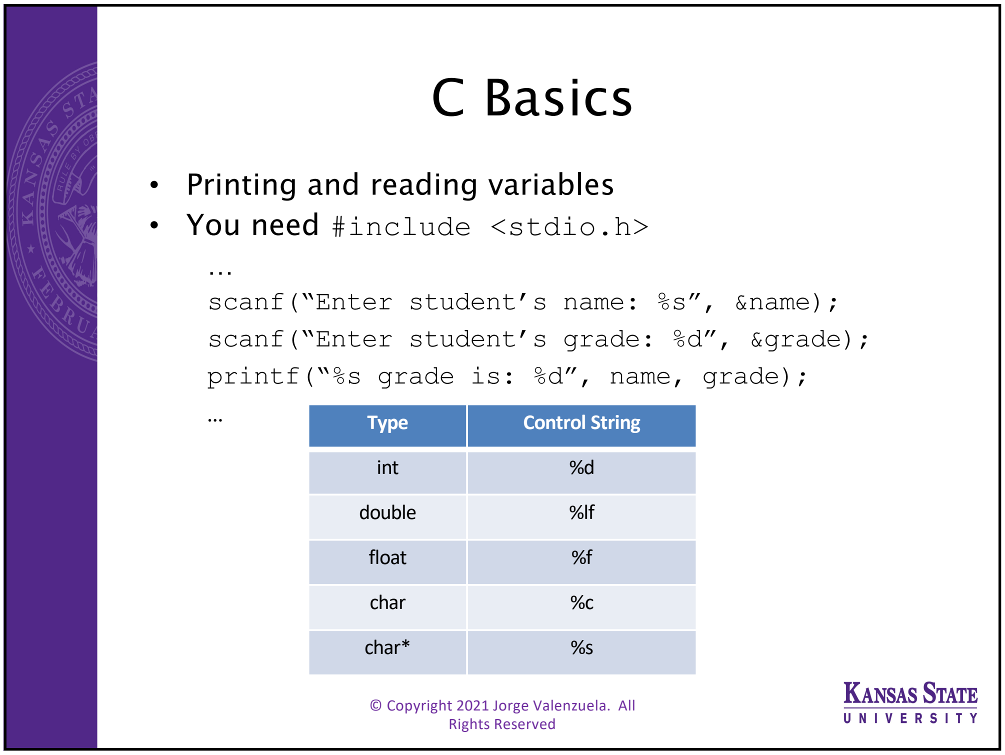
Global

```
int readInput();

int main() {
   int inNum = readInput();
   if(even) {
      printf("You entered %d, but we incremented to: %d\n", inNum/increment, inNum);
   } else {
      printf("You entered %d\n", inNum);
   }
   return 0;
}
int readInput() {
   int num;
   printf("Enter a number: \n");
   scanf("%d", &num); getchar();
   if(num % 2 == 0) {
      even = 1; // true
      num = num * increment;
   }
   else {
      even = 0; // false
   }
   return num;
}
```

Local

KANSAS STATE UNIVERSITY

19



C Basics

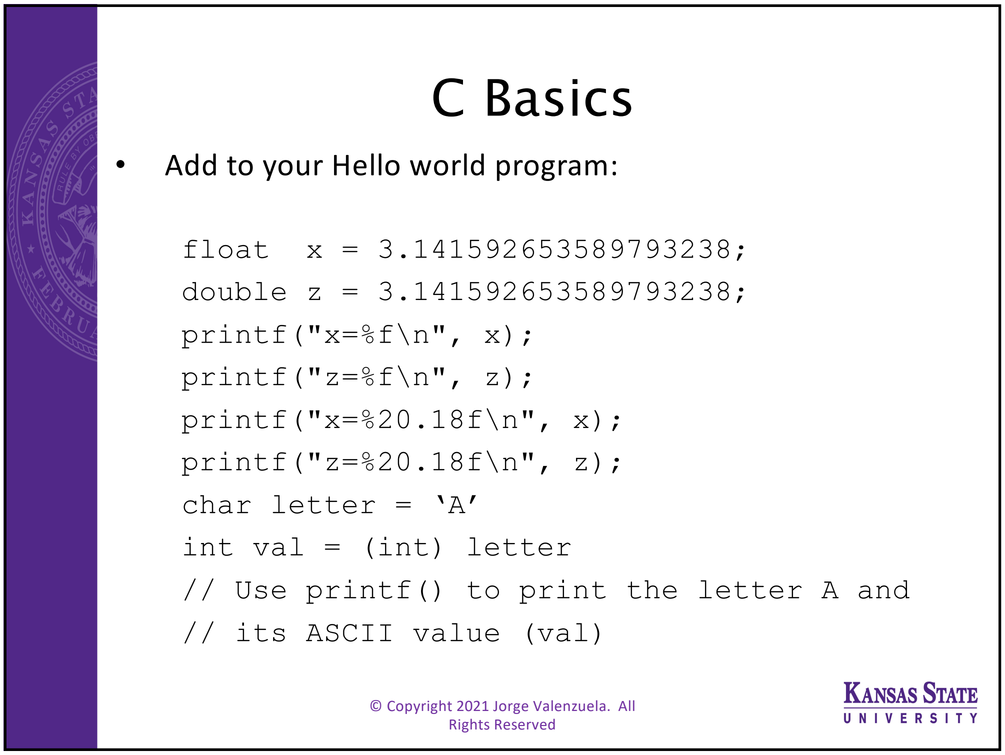
- Printing and reading variables
- You need `#include <stdio.h>`

```
…
scanf("Enter student's name: %s", &name);
scanf("Enter student's grade: %d", &grade);
printf("%s grade is: %d", name, grade);
…
```

Type	Control String
int	%d
double	%lf
float	%f
char	%c
char*	%s

KANSAS STATE UNIVERSITY

20

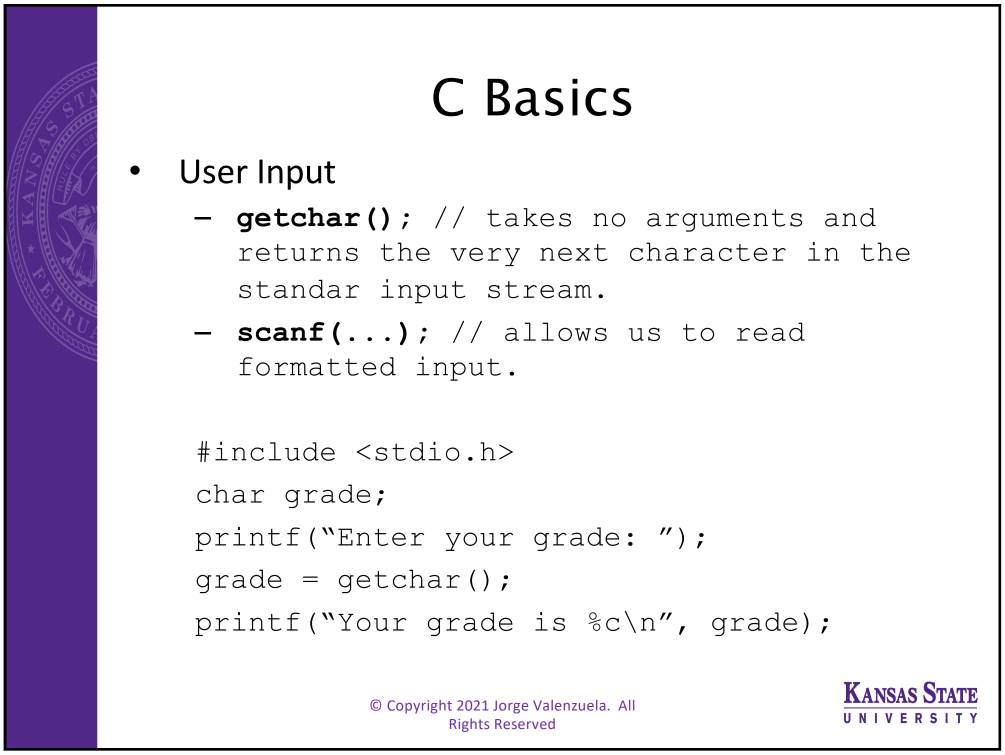


C Basics

- Add to your Hello world program:

```
float  x = 3.141592653589793238;
double z = 3.141592653589793238;
printf("x=%f\n", x);
printf("z=%f\n", z);
printf("x=%20.18f\n", x);
printf("z=%20.18f\n", z);
char letter = ‘A’
int val = (int) letter
// Use printf() to print the letter A and
// its ASCII value (val)
```

21

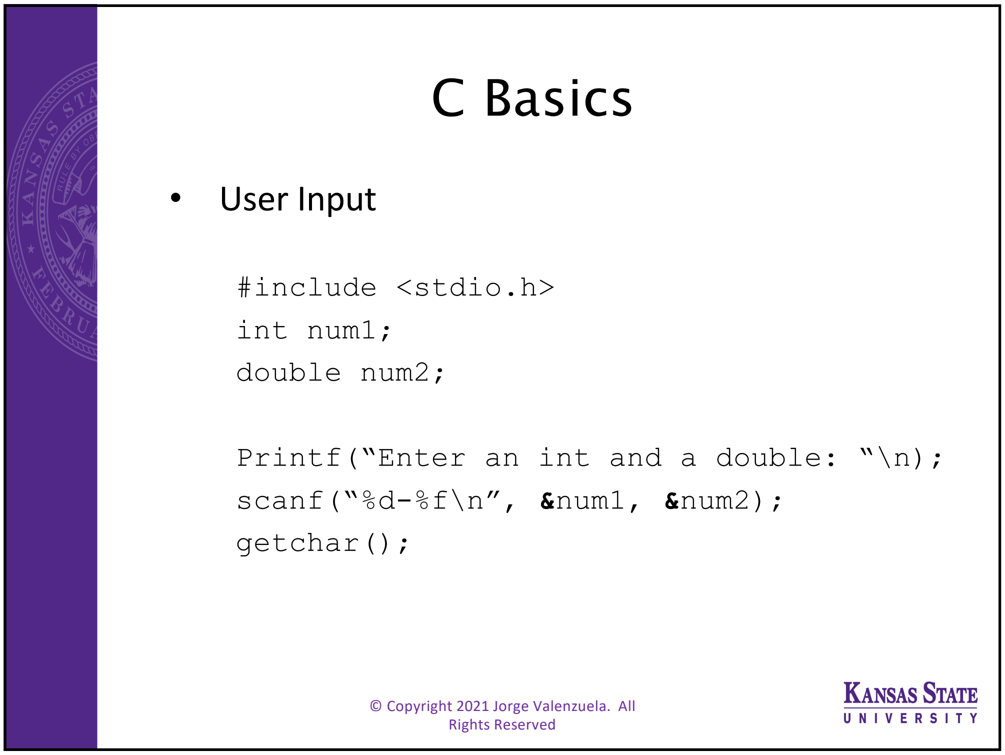


C Basics

- User Input
  - **getchar()**; // takes no arguments and returns the very next character in the standar input stream.
  - **scanf(...)**; // allows us to read formatted input.

```
#include <stdio.h>
char grade;
printf(“Enter your grade: ”);
grade = getchar();
printf(“Your grade is %c\n”, grade);
```

22

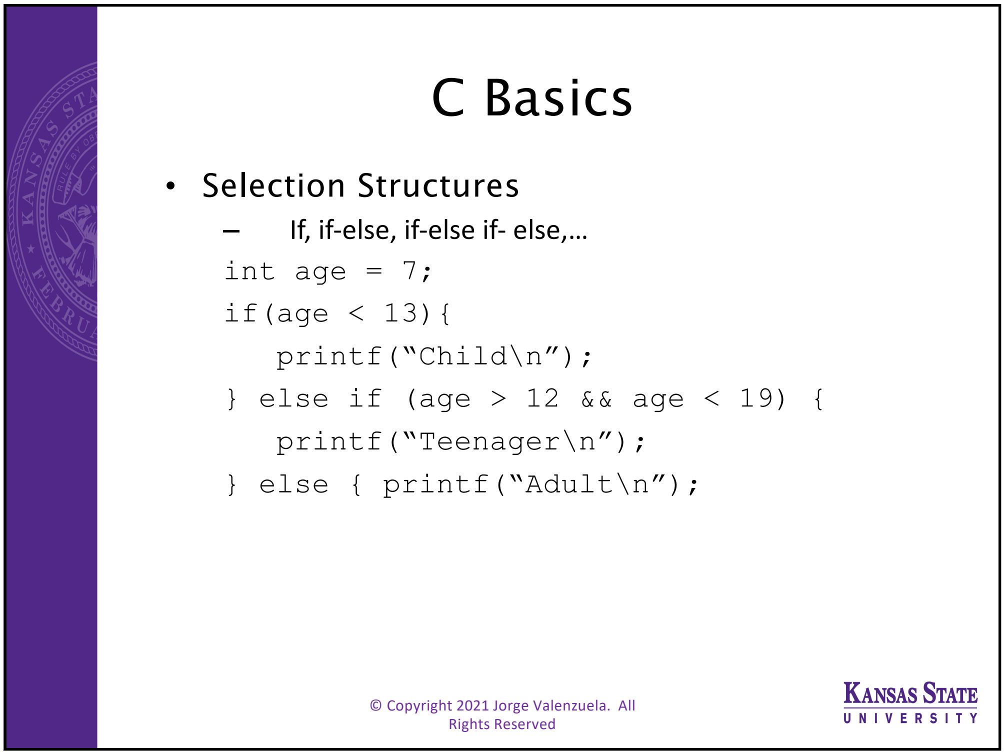


C Basics

- User Input

```
#include <stdio.h>
int num1;
double num2;

Printf(“Enter an int and a double: “\n);
scanf(“%d-%f\n”, &num1, &num2);
getchar();
```

23

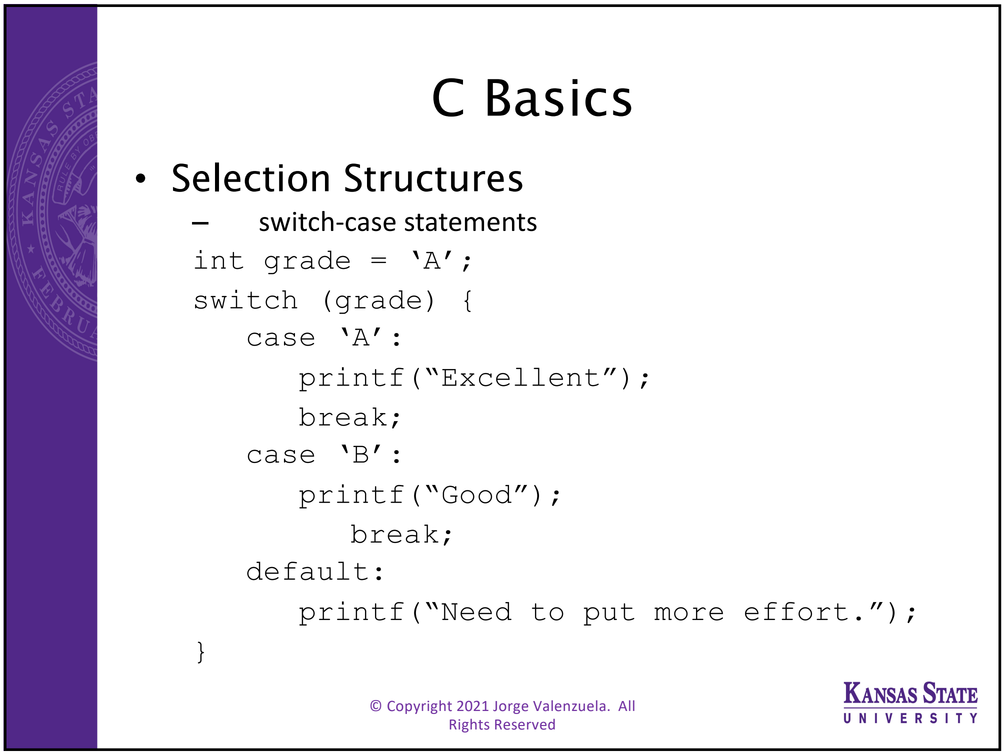


C Basics

- Selection Structures
  - If, if-else, if-else if- else,…

```
int age = 7;
if(age < 13){
   printf(“Child\n”);
} else if (age > 12 && age < 19) {
   printf(“Teenager\n”);
} else { printf(“Adult\n”);
```

24



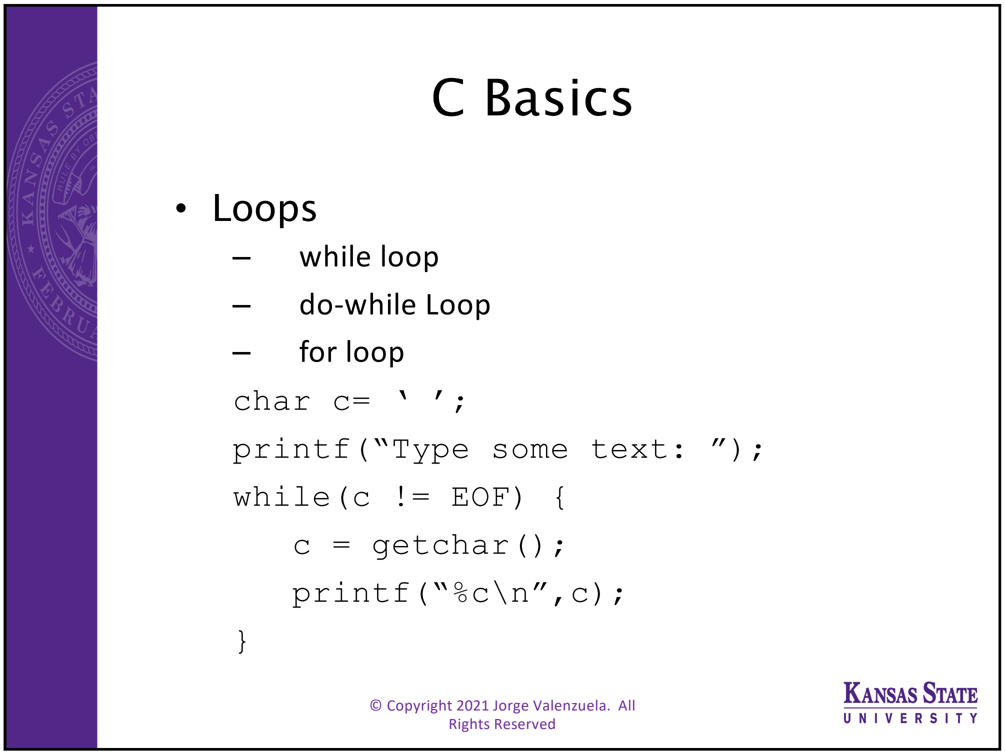
C Basics

- Selection Structures
  - switch-case statements

```
int grade = ‘A’;
switch (grade) {
   case ‘A’:
      printf(“Excellent”);
      break;
   case ‘B’:
      printf(“Good”);
         break;
   default:
      printf(“Need to put more effort.”);
}
```

KANSAS STATE UNIVERSITY

25



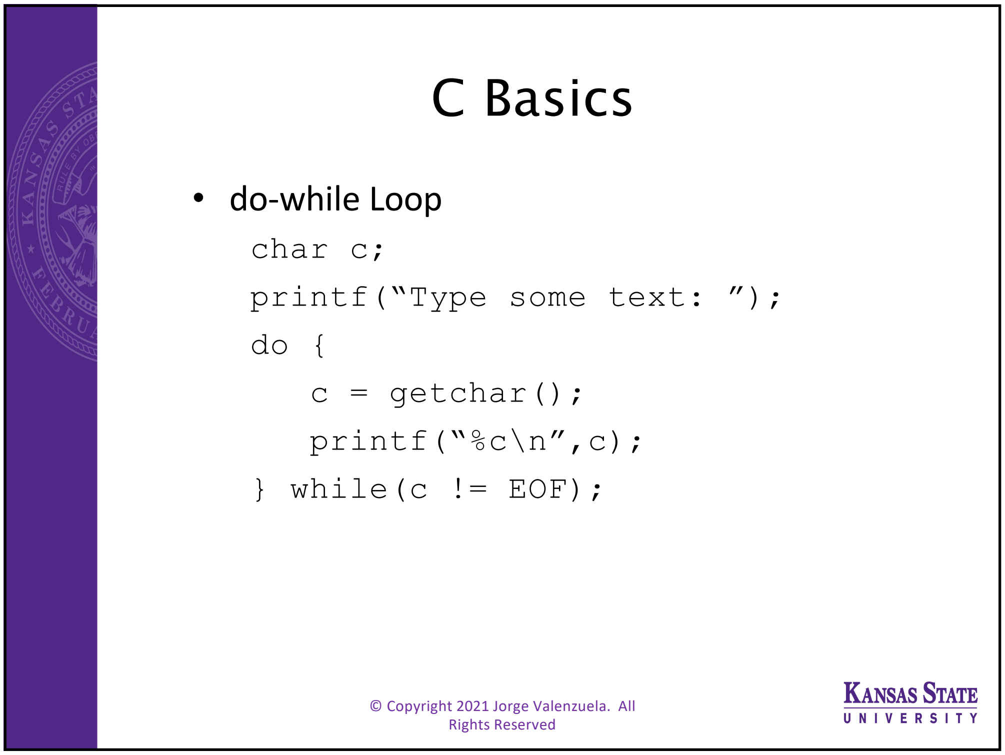
C Basics

- Loops
  - while loop
  - do-while Loop
  - for loop

```
char c= ‘ ’;
printf(“Type some text: ”);
while(c != EOF) {
   c = getchar();
   printf(“%c\n”,c);
}
```

KANSAS STATE UNIVERSITY

26

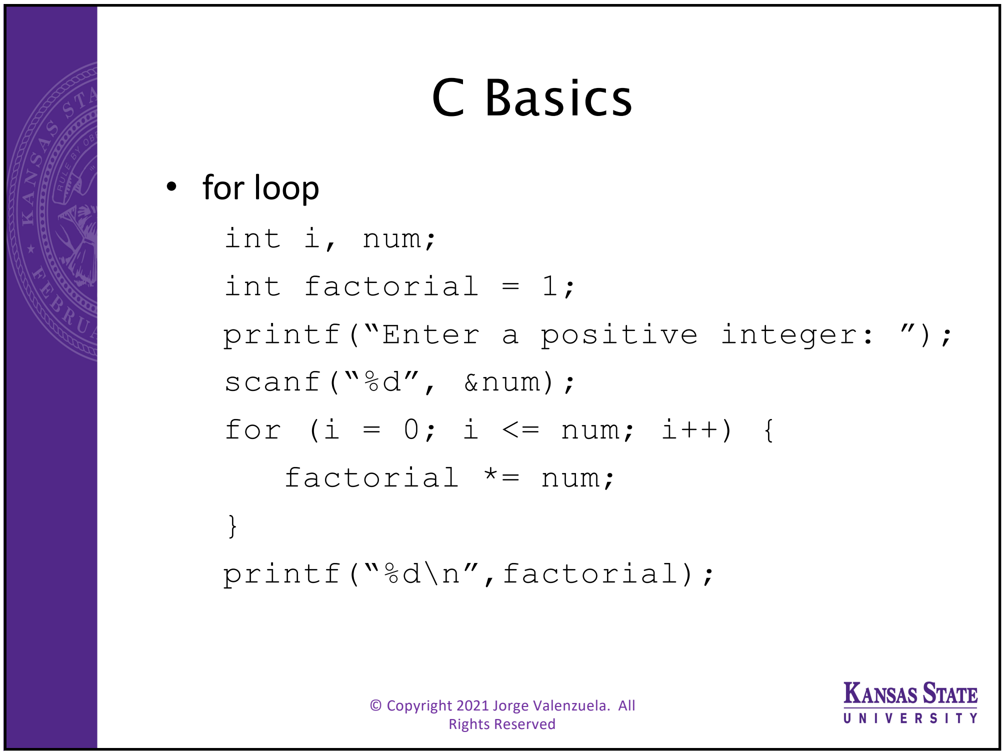


C Basics

- do-while Loop

```
char c;
printf(“Type some text: ”);
do {
   c = getchar();
   printf(“%c\n”,c);
} while(c != EOF);
```

27

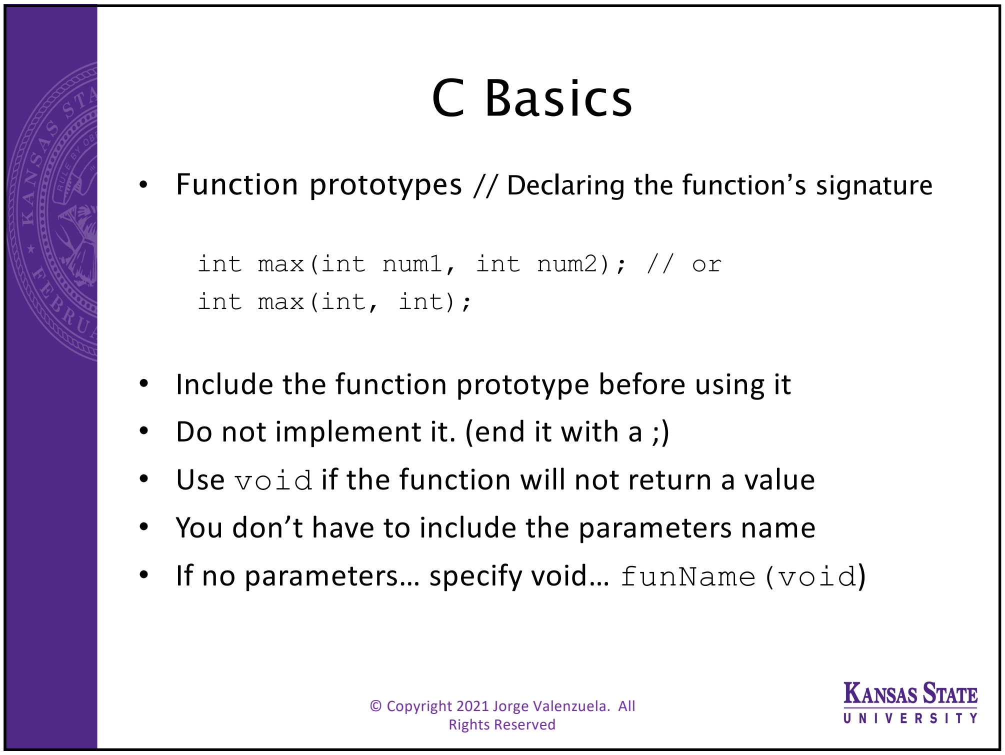


C Basics

- for loop

```
int i, num;
int factorial = 1;
printf(“Enter a positive integer: ”);
scanf(“%d”, &num);
for (i = 0; i <= num; i++) {
   factorial *= num;
}
printf(“%d\n”,factorial);
```

28



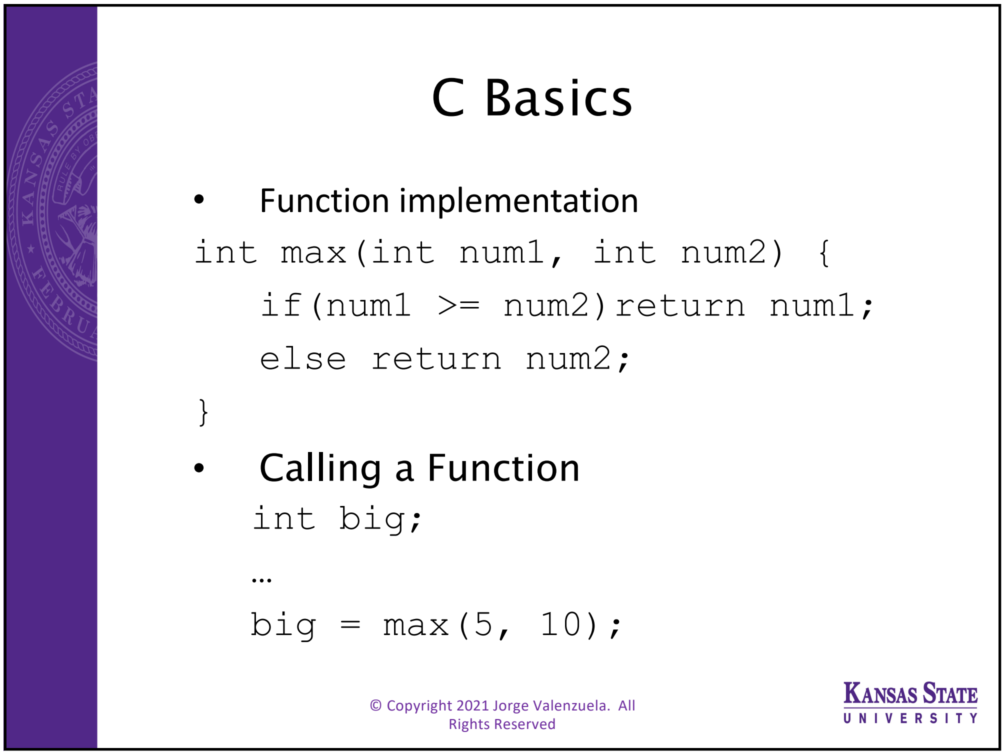
C Basics

- Function prototypes // Declaring the function’s signature

```
int max(int num1, int num2); // or
int max(int, int);
```

- Include the function prototype before using it
- Do not implement it. (end it with a ;)
- Use `void` if the function will not return a value
- You don’t have to include the parameters name
- If no parameters… specify void… `funName(void)`

29



C Basics

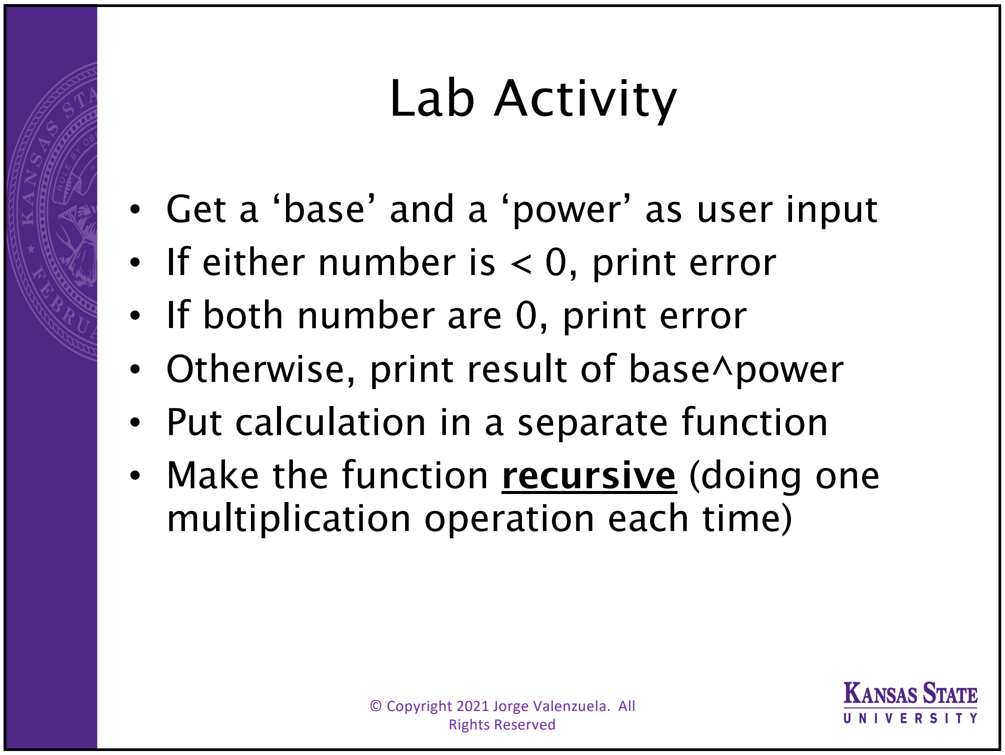
- Function implementation

```
int max(int num1, int num2) {
   if(num1 >= num2)return num1;
   else return num2;
}
```

- Calling a Function

```
int big;
…
big = max(5, 10);
```

30



Lab Activity

- Get a ‘base’ and a ‘power’ as user input
- If either number is < 0, print error
- If both number are 0, print error
- Otherwise, print result of base^power
- Put calculation in a separate function
- Make the function **recursive** (doing one multiplication operation each time)

KANSAS STATE
UNIVERSITY

31