

# Scope

Last updated 10/29/20

# Scope

- These slides introduce the concept of scope
- Upon completion: You should be able to interpret and use scope with variables and functions

# Scope

- Scope
  - Region of a program in which a defined object is visible
  - Defined Objects
    - Variables
    - Functions
  - Two types of regions
    - Blocks
    - Not in a block

# Scope

- Program Prototype

- Blocks

- Statements enclosed in { ... }

- Contents of Main
- Contents of Functions

- Not in a Block

- Global Area

```
// comments
```

```
#include <stdio.h>  
int foo;
```

Global Area

```
int fun1(int x, int y); // function prototype
```

```
int main(void){
```

```
int x;  
int y;  
float a;  
if(...){
```

Main's Area

```
float x;  
float a;  
float b;  
x = a * 3  
}
```

Nested Block Area

```
else  
    a = x * y;  
...  
} // end of main
```

```
int fun1 (int i, int j){
```

```
int x;  
int y;
```

Function fun1 Area

```
...  
} // end of fun1
```

# Scope

- Scope
  - An objects scope extends from it's declaration to the end of it's block
  - Global Scope
    - Any object defined in the global area of a program
    - Visible anywhere in the current program
  - Local Scope
    - Any object defined in a block area
    - Includes Main and Functions
    - Visible anywhere in the current block

# Scope

- Scope
  - Local definitions supersede global definitions within a block

```
// example
```

```
#include <stdio.h>
```

```
int x;
```

```
int y;
```

```
int main(void){
```

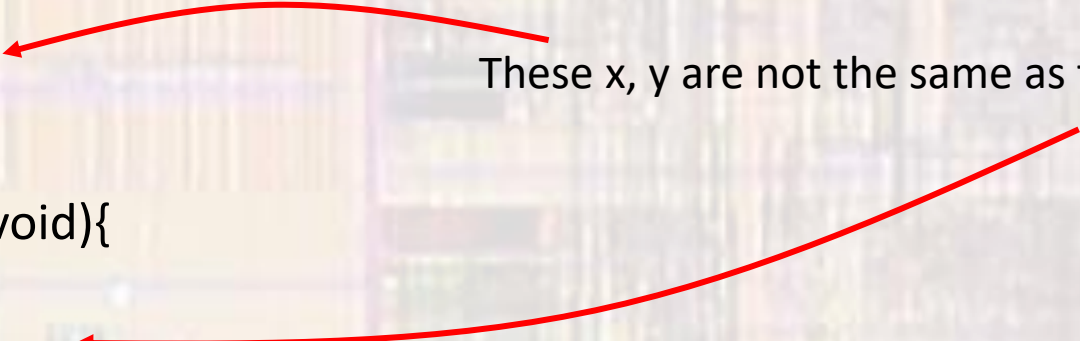
```
    int x;
```

```
    float y;
```

```
    ...
```

```
}
```

These x, y are not the same as these



# Scope

- Scope

```
// comments
#include <stdio.h>
int foo;

int fun1(int x, int y);    // function prototype

int main(void){
  int x;
  int y;
  float a;
  if(...){
    float x;
    x = a * 3;
    float a;
    float b;
  }
  else
    b = x * y;
  ...
} // end of main

int fun1 (int i, int j){
  int x;
  int y;
  ...
} // end of fun1
```

The diagram illustrates variable scope in C. It shows a code snippet with several variables and annotations. Arrows point from the annotations to the corresponding variable declarations in the code. The annotations are: 'foo is visible here' pointing to the global 'foo' declaration; 'this a is visible here but this is a new a' pointing to the 'float a;' declarations in 'main' and the nested 'if' block; 'new i,j new x,y only visible in fun1' pointing to the 'int i, int j', 'int x', and 'int y' declarations in the 'fun1' function.

foo is visible here

this a is visible here  
but  
this is a new a

new i,j  
new x,y  
only visible in fun1

# Scope

- Scope

```
/* vegas.c
 *
 * Created on: Sep 21, 2016
 * Author: Tim
 */

// scope of variables illustration
// Copyright by Kerry R. Widder
// 9/15/16

#include <stdio.h>

int vegas(int i, int j);

int main(void){
    setbuf(stdout, NULL);
    int i;
    int j;
    int k;
    i = 2;
    j = 4;
    k = 0;
    printf("i = %i, j = %i, k = %i \n", i, j, k);
    k = vegas(i, j);
    printf("i = %i, j = %i, k = %i \n", i, j, k);

    return 0;
} // end main

int vegas(int i, int j){
    int new;
    new = 0;
    i++;
    j--;
    new = i * j;
    return new;
} // end vegas
```



# Scope

- Scope

```
/* vegas.c
 *
 * Created on: Sep 21, 2016
 * Author: Tim
 */

// scope of variables illustration
// Copyright by Kerry R. Widder
// 9/15/16

#include <stdio.h>

int vegas(int i, int j);

int main(void){
    setbuf(stdout, NULL);
    int i;
    int j;
    int k;
    i = 2;
    j = 4;
    k = 0;
    printf("i = %i, j = %i, k = %i \n", i, j, k);
    k = vegas(i, j);
    printf("i = %i, j = %i, k = %i \n", i, j, k);

    return 0;
} // end main

int vegas(int i, int j){
    int new;
    new = 0;
    i++;
    j--;
    new = i * j;
    return new;
} // end vegas
```

```
i = 2, j = 4, k = 0
i = 2, j = 4, k = 9
```

# Scope

- Static Variables
  - Hold their value even after their scope has ended

```
* static_ex.c
*
* Created on: Jan 20, 2020
* Author: johnsontimoi
*/
////////////////////////////////////
//
// example of a static variable holding its value
//
////////////////////////////////////

#include <stdio.h>

int fun1(void);
int fun2(void);

int main(void){
    printf("%d ", fun1());
    printf("%d ", fun1());
    printf("%d ", fun2());
    printf("%d ", fun2());

    return 1;
} // end main

int fun1(void){
    int count;
    count = 0;
    count++;
    return count;
} // end fun1

int fun2(void){
    static int count = 0; // special case for assignment
    count++;
    return count;
} // end fun2
```

<terminated>

1 1 1 2