Scope

Last updated 6/16/23

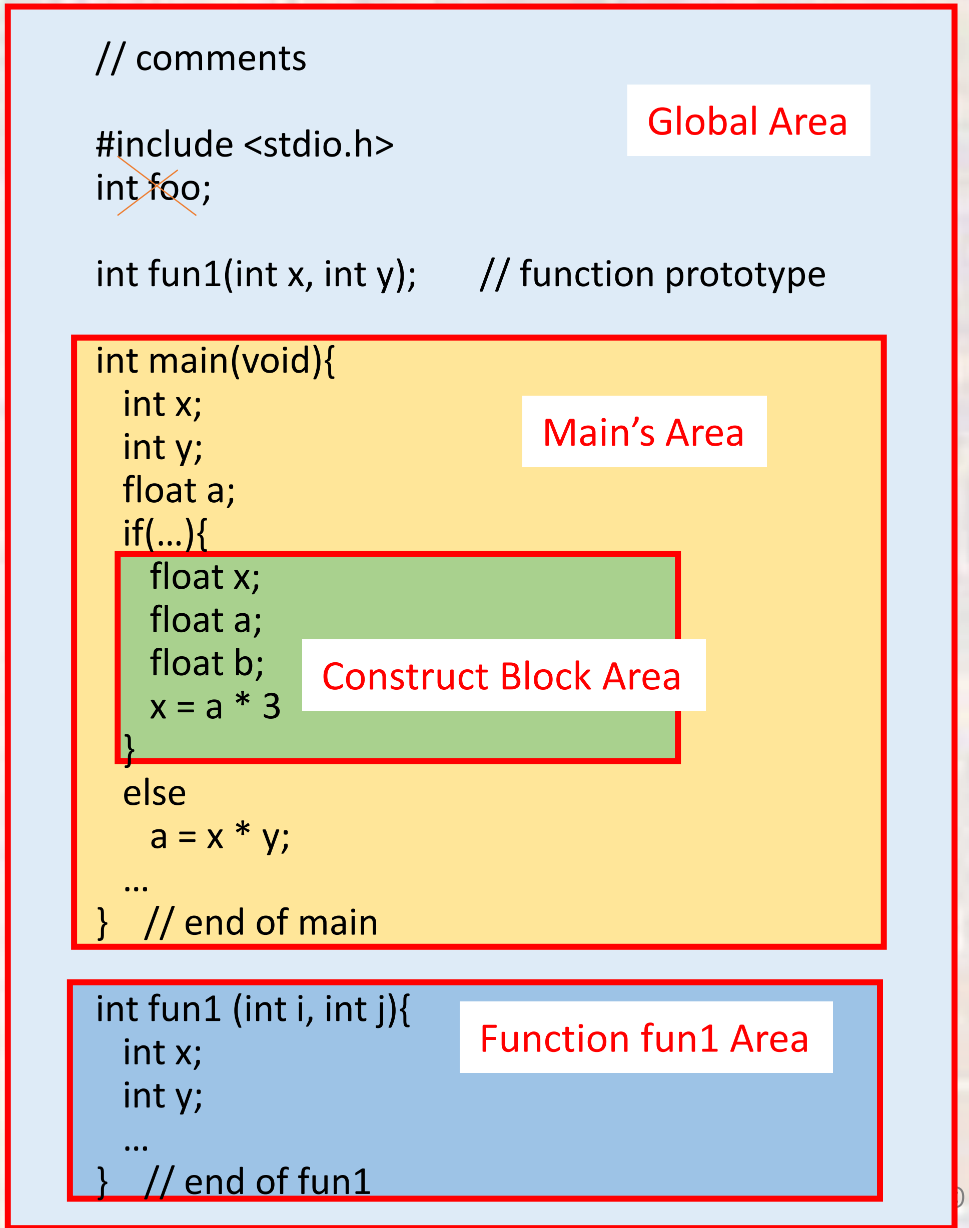
These slides describe scope in C

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
    - Contents of Constructs
  - Not in a Block
    - Global Area



```
// 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;
    float a;
    float b;
    x = a * 3
  }
  else
    a = x * y;
  …
}   // end of main

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

Scope

- Regions
  - 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
  - Active scope is prioritized from inside → out

Scope

- Regions

  - 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

- Example

```
// 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
```

foo is visible here

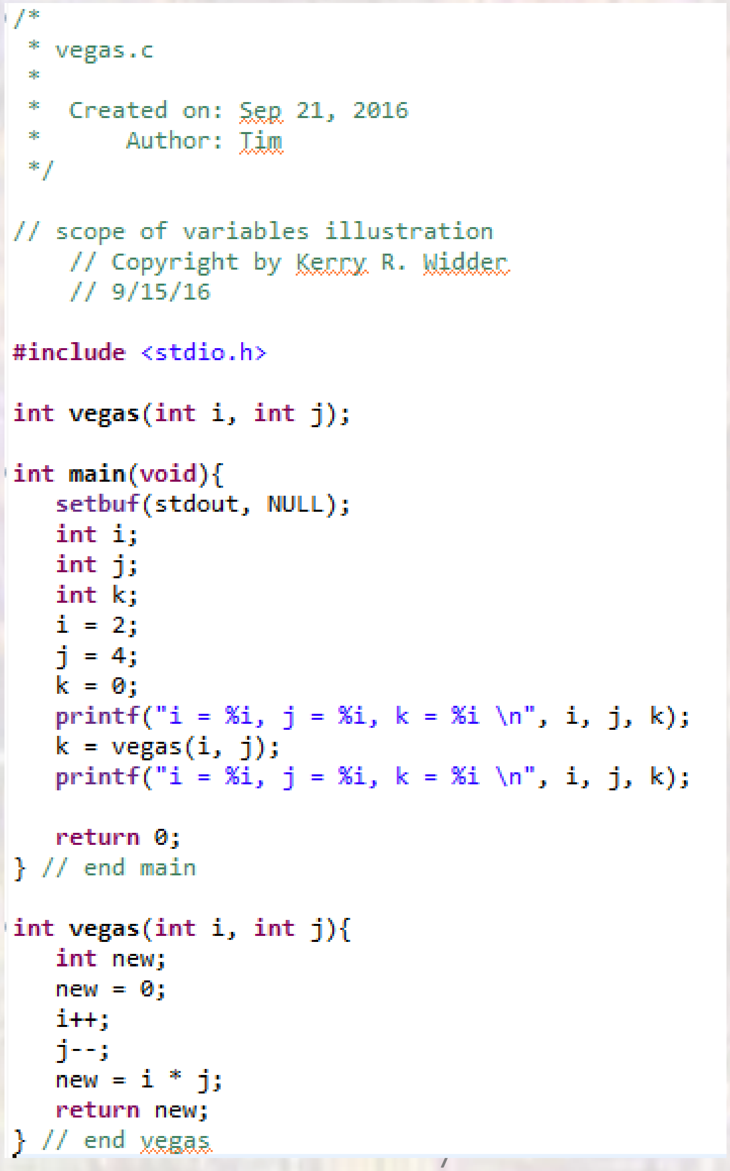
this a is visible here
but
this is a new a

new i,j
new x,y

only visible in fun1

Scope

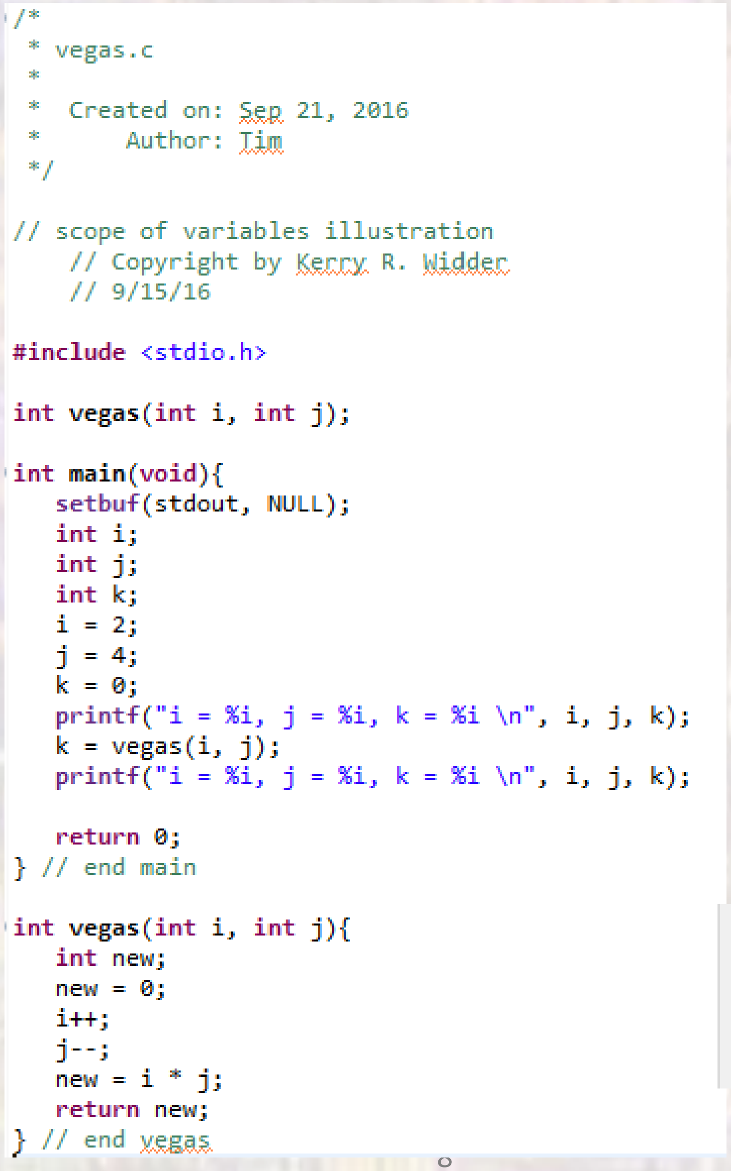
- Example



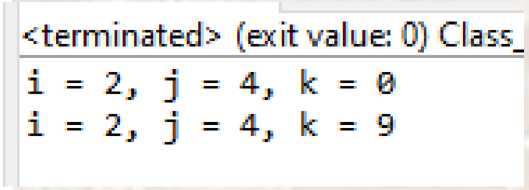
```
/*
 * vegas.c
 *
 *  Created on: Sep 21, 2016
 *      Author: Tim
 */

// 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

- Example



```
/*
 * 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
```



```
<terminated> (exit value: 0) Class_
i = 2, j = 4, k = 0
i = 2, j = 4, k = 9
```

Scope

- Lesson Learned
  - It is easy to get confused if you use the same variable names in different scopes
    - Especially in function definitions
    - Best practice – use unique names for function FORMAL parameters
  - Remember – the values of variables are passed to functions, NOT the variables themselves

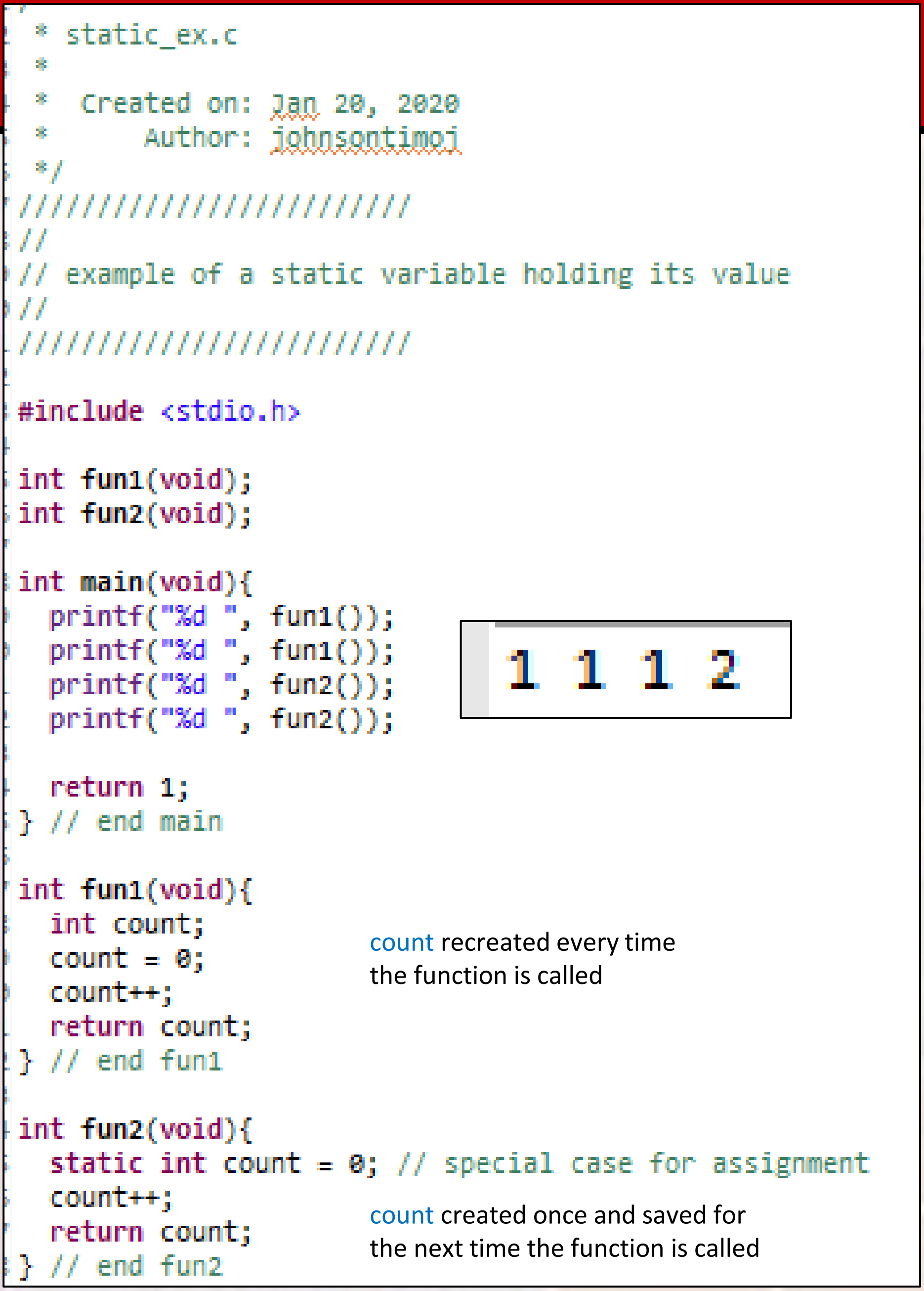
```
int volume(int L, int W, int H);

int main(void){
…
total = volume(length, width, height);
…
```

```
float savings(float bal, float int_rate);

int main(void){
…
interest = savings(current_balance, interest_rate);
…
```

```
float ave(float val1, float val2);

int main(void){
…
average = ave(a, b);
…
```

Scope

- Static Variables
  - Hold their value inside a scope even after their scope has ended
  - Often used in functions
  - Stored separately in Data Memory



```
 * static_ex.c
 *
 *  Created on: Jan 20, 2020
 *      Author: johnsontimoj
 */
/////////////////////////
//
// 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
```

```
1 1 1 2
```

count recreated every time the function is called

count created once and saved for the next time the function is called