Mahidol University

ITCS113 Fundamentals of Programming

Lecture 10 - Variable Scope

Instructor: Asst. Prof. Dr. Akara Supratak

Contact: akara.sup@mahidol.edu

Wisdom of the Land

Agenda

- Function Scope
- Variable Scope
- Guideline on Declaring Variables

Mahidol University

Function Scope

Function Scope: Blocks

Every function has its own scope (block)

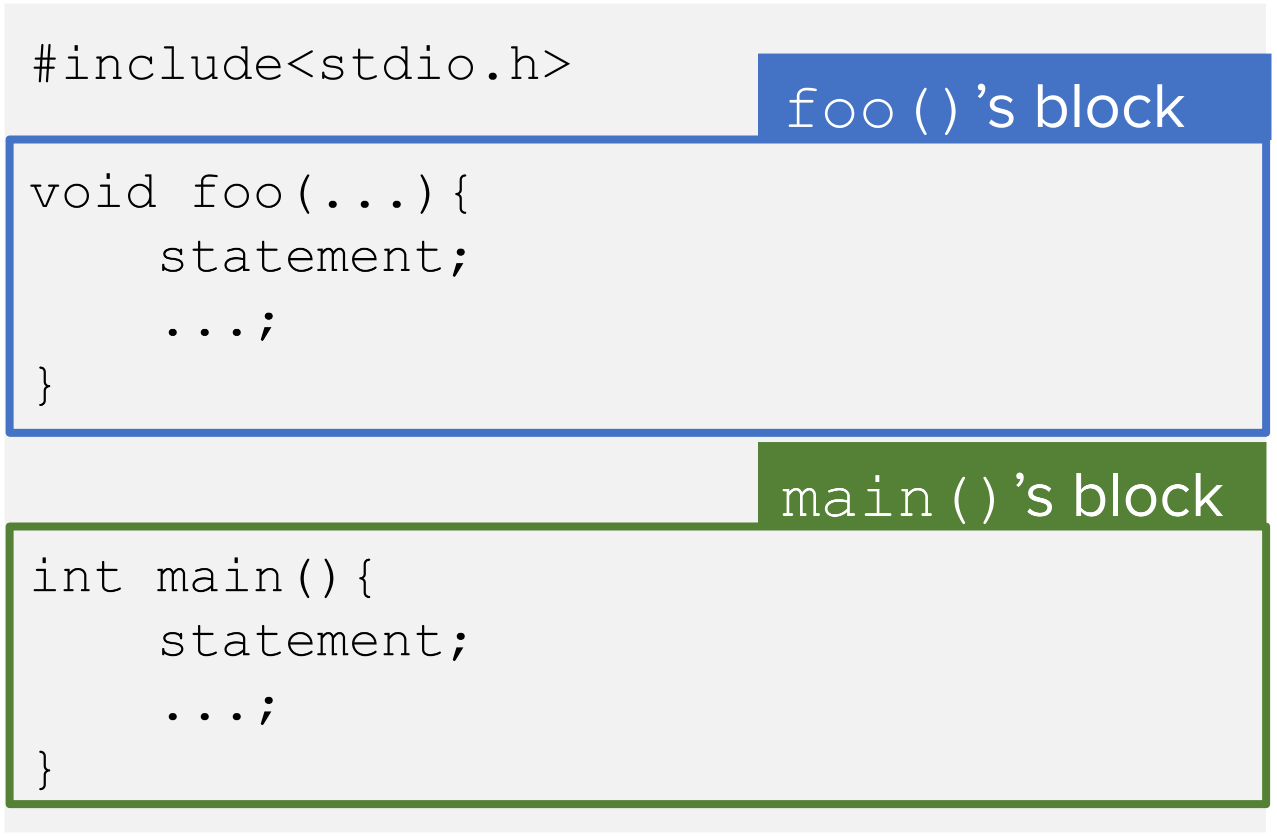
```
#include<stdio.h>

void foo(...){
    statement;
    ...;
}


int main(){
    statement;
    ...;
}
```

Function Scope: Blocks

Every function has its own scope (block)



```
#include<stdio.h>

void foo(...){
    statement;
    ...;
}

int main(){
    statement;
    ...;
}
```

Function Scope: Variables

- Every variable inside the function is only usable during the execution of the function
- The variables are destroyed at the end of the function

```
#include<stdio.h>

void foo(){
    int o = 100;
}

int main(){
    foo();
    printf("%d", o);
    return 0;
}
```

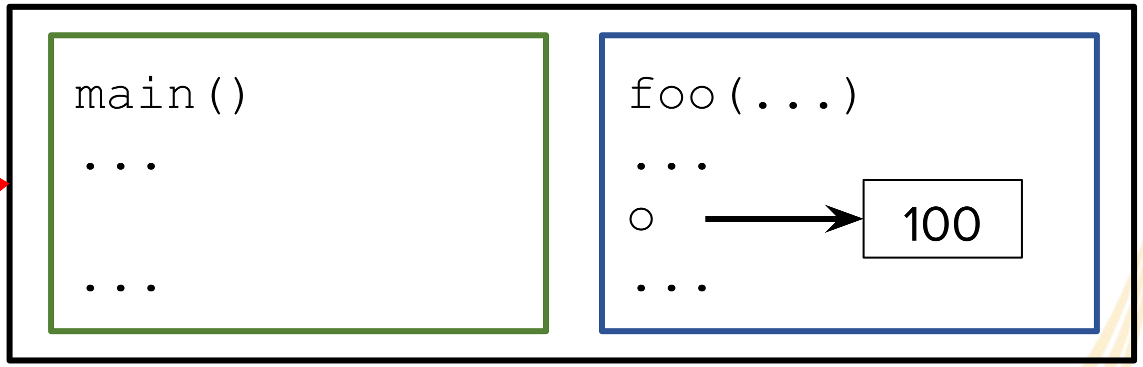
Function Scope: Variables

- Every variable inside the function is only usable during the execution of the function
- The variables are destroyed at the end of the function

```
#include<stdio.h>

void foo(){
    int o = 100;
}

int main(){
    foo();
    printf("%d", o);
    return 0;
}
```



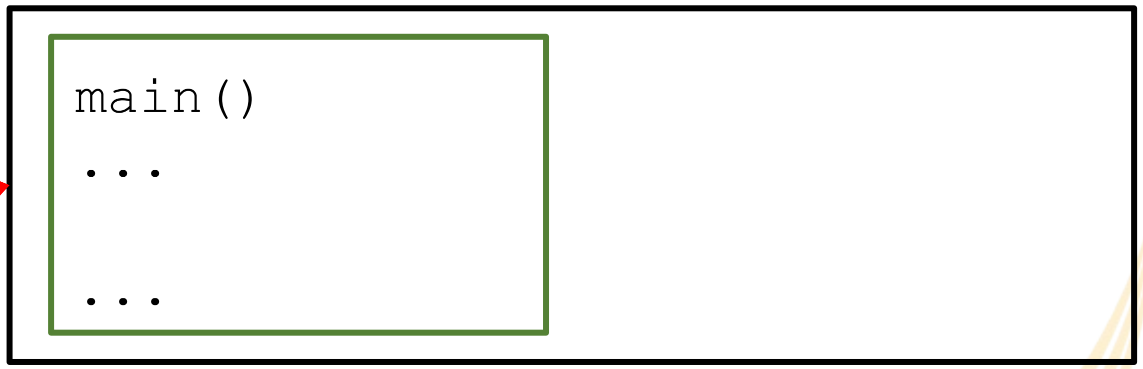
Function Scope: Variables

- Every variable inside the function is only usable during the execution of the function
- The variables are destroyed at the end of the function

```
#include<stdio.h>

void foo(){
    int o = 100;
}

int main(){
    foo();
    printf("%d", o);
    return 0;
}
```



Function Scope: Variables

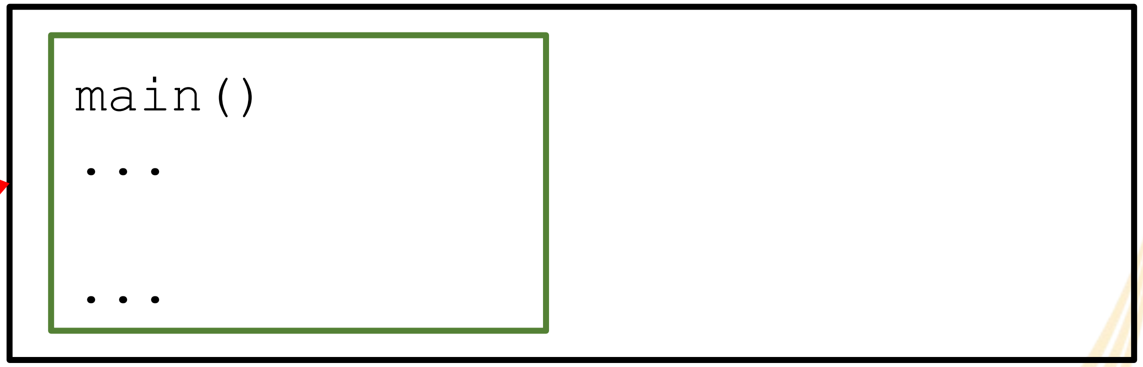
- Every variable inside the function is only usable during the execution of the function
- The variables are destroyed at the end of the function

```
#include<stdio.h>

void foo(){
    int o = 100;
}

int main(){
    foo();
    printf("%d", o);
    return 0;
}
```

Error !!!



Function Scope: Variables

Functions cannot access variable in other functions

```
#include<stdio.h>

void foo(){
    o++;
}

int main(){
    int o = 10;
    foo();
    return 0;
}
```

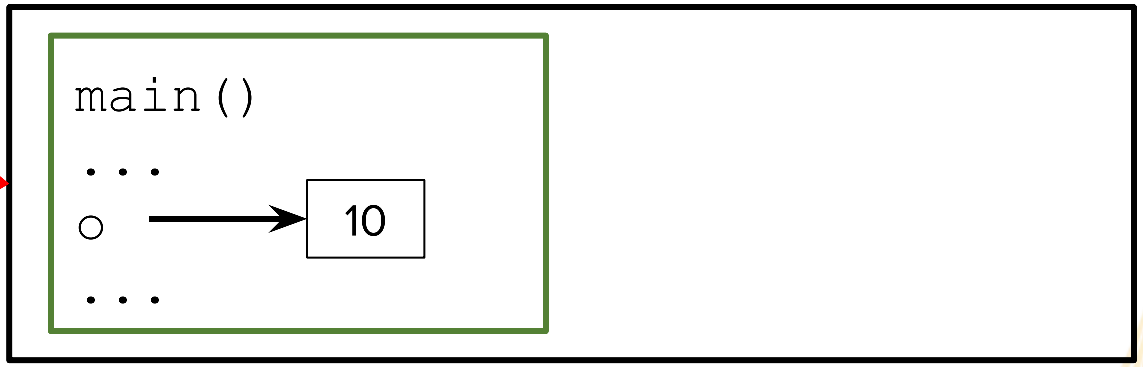
Function Scope: Variables

Functions cannot access variable in other functions

```
#include<stdio.h>

void foo(){
    o++;
}

int main(){
    int o = 10;
    foo();
    return 0;
}
```



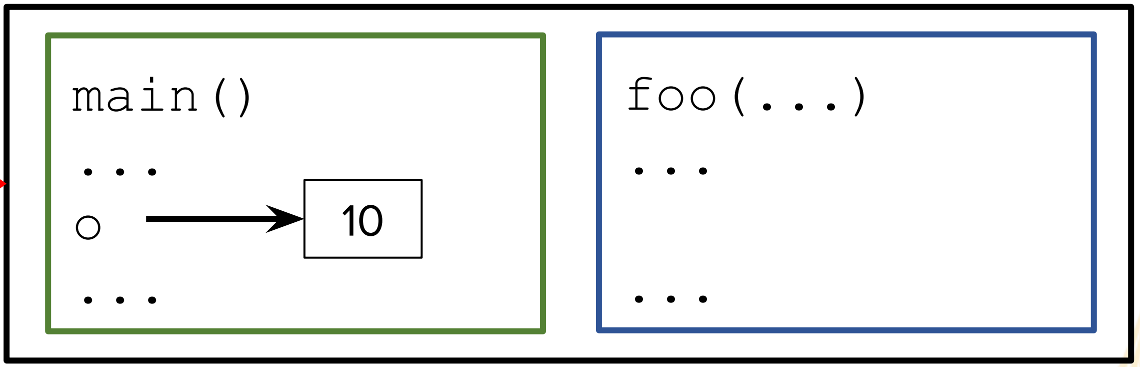
Function Scope: Variables

Functions cannot access variable in other functions

```
#include<stdio.h>

void foo(){
    o++;
}

int main(){
    int o = 10;
    foo();
    return 0;
}
```



Function Scope: Variables

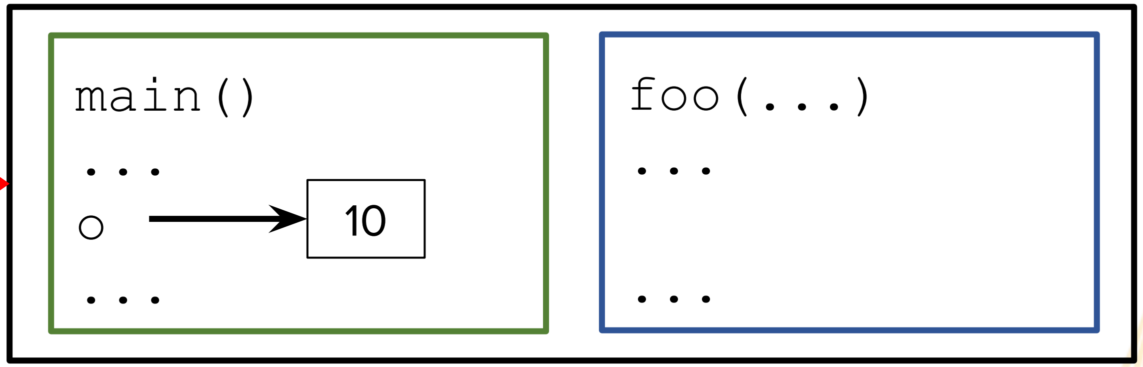
Functions cannot access variable in other functions

```
#include<stdio.h>

void foo(){
    o++;
}

int main(){
    int o = 10;
    foo();
    return 0;
}
```

Error !!!



Function Scope: Variables

Function parameters behave like **local** variables of the function

```
#include<stdio.h>

void foo(int x){
    printf("%d", x);
}

int main(){
    foo(42);
    printf("%d", x);
    return 0;
}
```

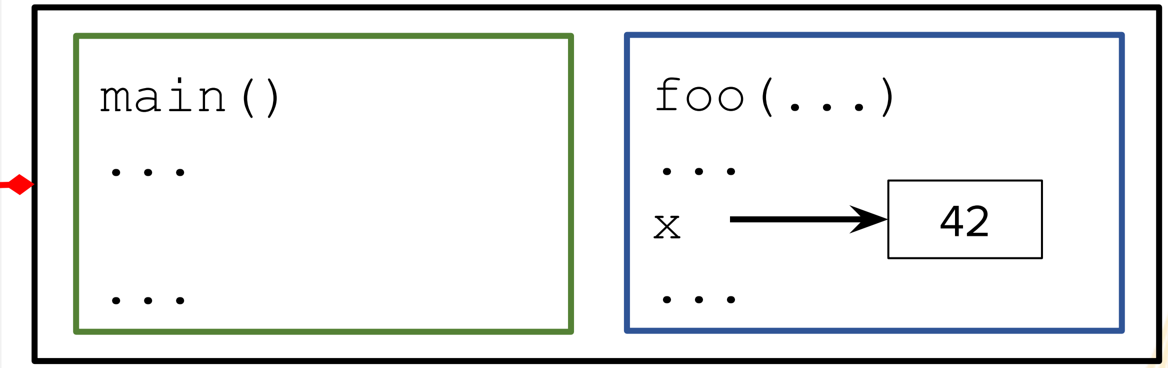
Function Scope: Variables

Function parameters behave like **local** variables of the function

```
#include<stdio.h>

void foo(int x){
    printf("%d", x);
}

int main(){
    foo(42);
    printf("%d", x);
    return 0;
}
```



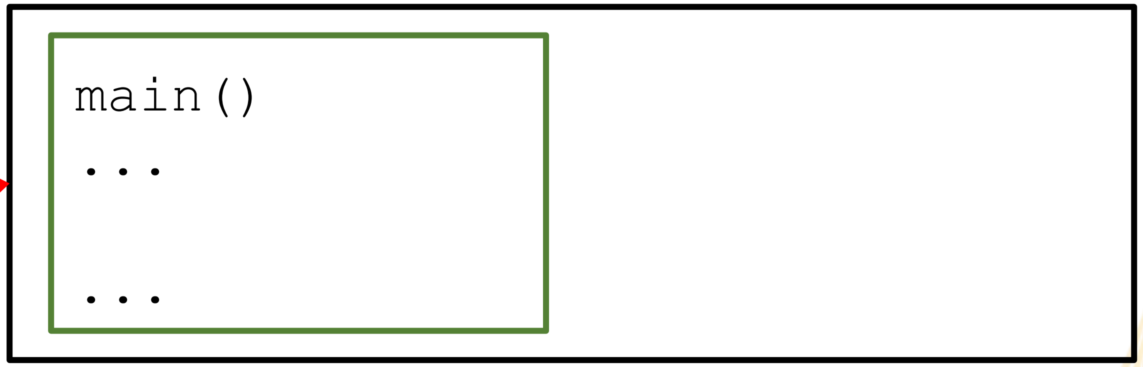
Function Scope: Variables

Function parameters behave like **local** variables of the function

```
#include<stdio.h>

void foo(int x){
    printf("%d", x);
}

int main(){
    foo(42);
    printf("%d", x);
    return 0;
}
```



Function Scope: Variables

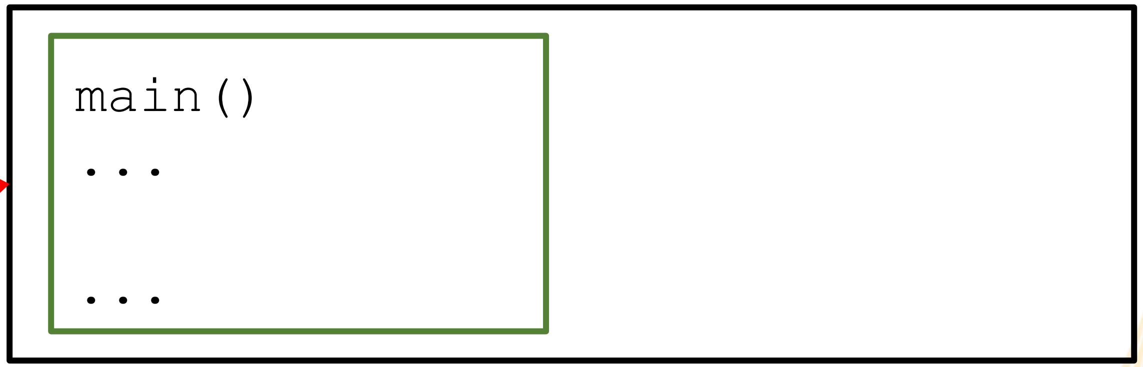
Function parameters behave like **local** variables of the function

```
#include<stdio.h>

void foo(int x){
    printf("%d", x);
}

int main(){
    foo(42);
    printf("%d", x);
    return 0;
}
```

Error !!!



Function Scope: Pass by Value

We usually pass just the “values”, not variables

```
#include<stdio.h>

void foo(int x, int y){
    y = 10;
    printf("%d %d", x, y);
}

int main(){
    int x = 1, y = 2;
    foo(y, x);
    printf("%d %d", x, y);
    return 0;
}
```

```
...;
foo(y, x);
    2  1
```

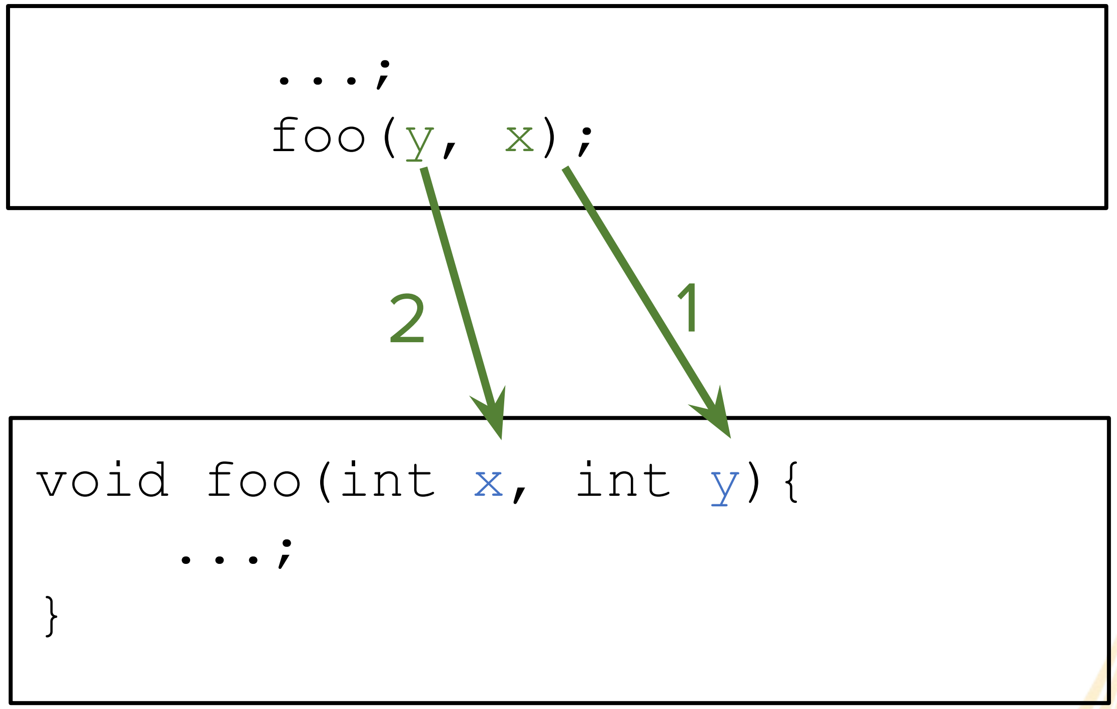
Function Scope: Pass by Value

We usually pass just the “values”, not variables

```c
#include<stdio.h>

void foo(int x, int y){
    y = 10;
    printf("%d %d", x, y);
}

int main(){
    int x = 1, y = 2;
    foo(y, x);
    printf("%d %d", x, y);
    return 0;
}
```



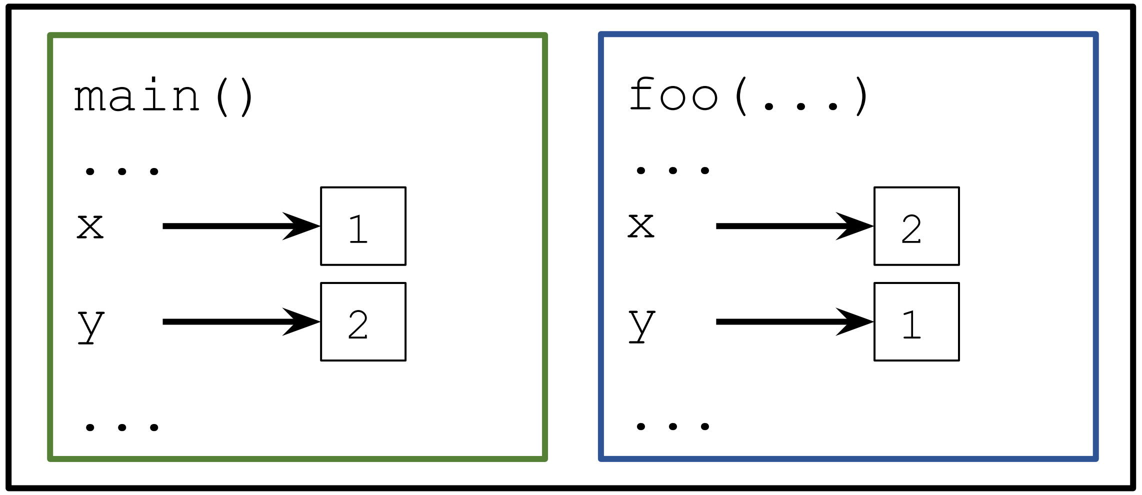
```c
        ...;
        foo(y, x);
```

```c
void foo(int x, int y){
    ...;
}
```

Function Scope: Pass by Value

We usually pass just the "values", not variables

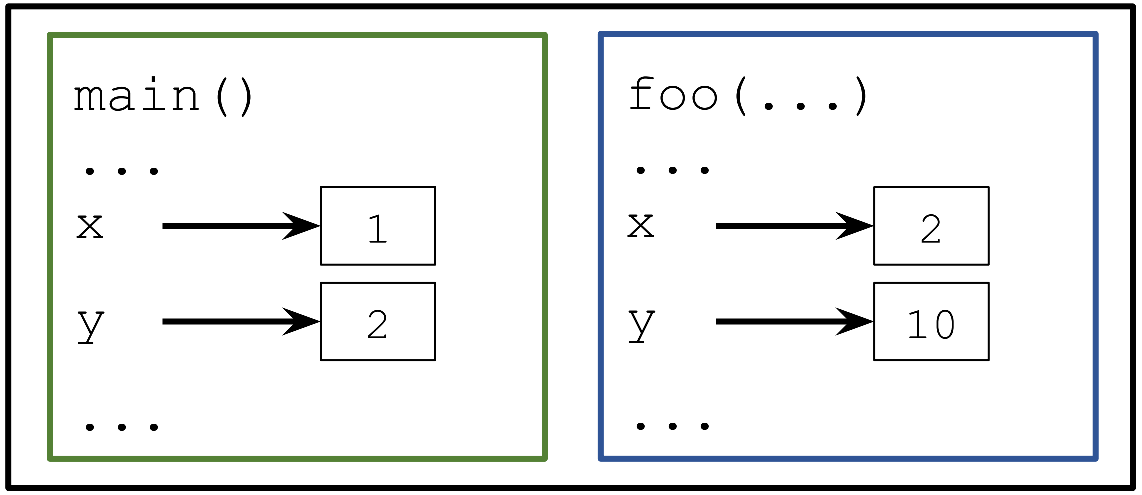
```
                 2        1
void foo(int x, int y){
    …
}
```



Function Scope: Pass by Value

We usually pass just the “values”, not variables

```
void foo(int x, int y){
    y = 10;
    …
}
```



Only change the value of the variable `y` defined <u>inside</u> the function

Exercise

What is the output?

```
/* #1 */
#include <stdio.h>

void foo(int x, int c) {
    printf("%d\n", x);
    x = 1000;
    c = 4;
    printf("%d\n", x);
}

int main() {
    int x = 42, d = 10;
    foo(x, d);
    printf("%d\n", x);
    printf("%d\n", d);
    return 0;
}
```

Exercise

What is the output?

```
/* #2 */
#include <stdio.h>

void foo(int x){
   printf("%d", y);
}

void main(){
   int y = 42;
   foo(y);
}
```

Exercise

What is the output?

```
/* #3 */
#include <stdio.h>

int is_odd(int i){
    if (i % 2 == 1) return 1;
    else return 0;
}

void foo(){
    int o = 1000;
}

int main(){
    int o = 0;
    for (int i = 0; i < 5; i++){
        if (is_odd(i)){
            o += i; // o = o + i
        }
    }
    foo();
    printf("%d\n", o);
    return 0;
}
```

Mahidol University

Variable Scope

Variable Scope

There are 3 places we can declare a variable.

1. Global Variable

2. Local Variable

3. Formal Parameters (i.e., local variable of the function or input arguments)

```
#include <stdio.h>

int foo;

int main()
{
    float x;
}

void foo(char param)
{
    …
}
```

Variable Scope

- A program is organized in blocks, called **Scope**
- Variables declared in a scope will only exist within the scope's boundary
- Variables in different scopes can have the same name

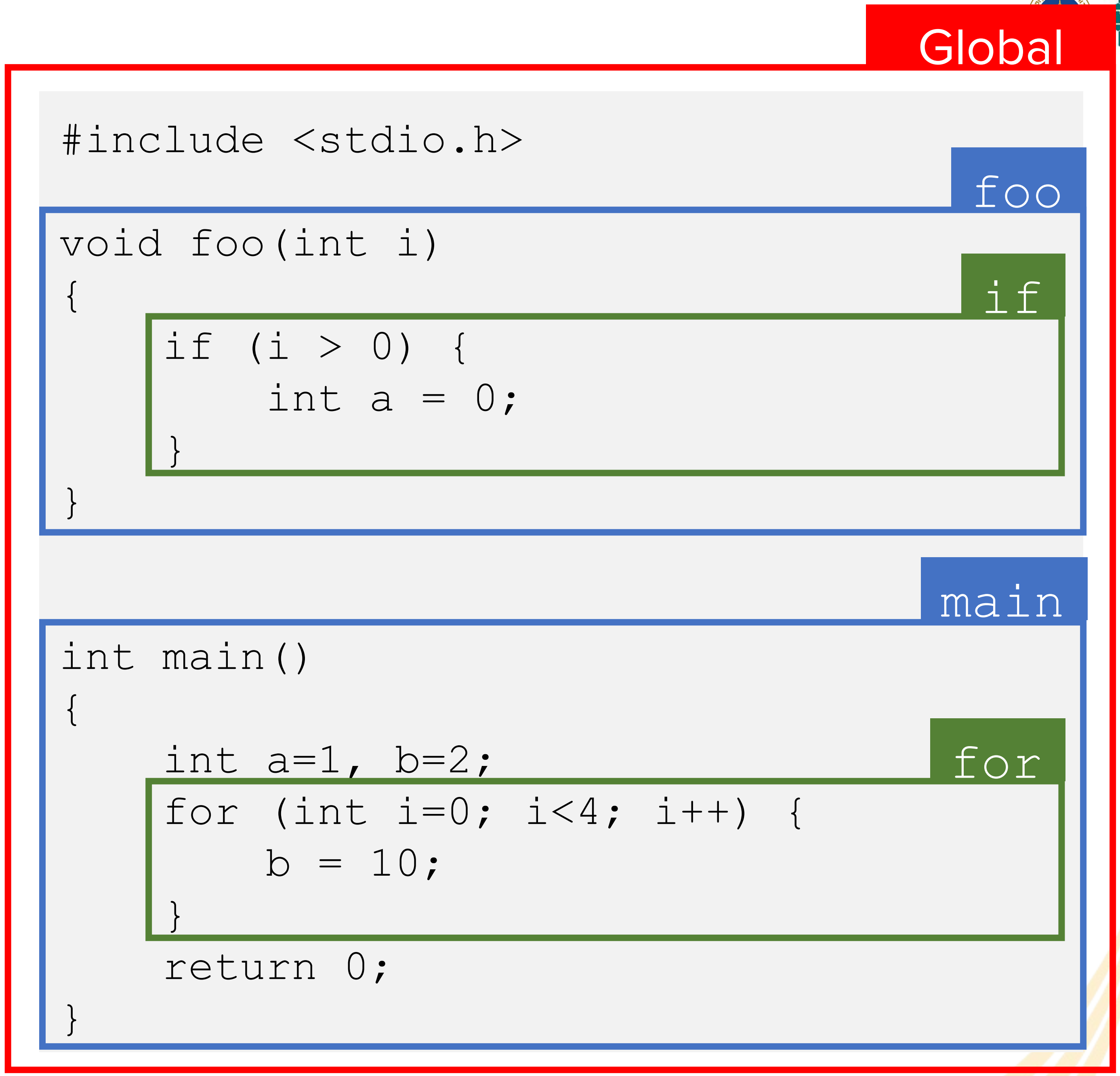
```
#include <stdio.h>

void foo(int i)
{
    if (i > 0) {
        int a = 0;
    }
}


int main()
{
    int a=1, b=2;
    for (int i=0; i<4; i++) {
        b = 10;
    }
    return 0;
}
```

Variable Scope

- A program is organized in blocks, called **Scope**
- Variables declared in a scope will only exist within the scope's boundary
- Variables in different scopes can have the same name



```
#include <stdio.h>

void foo(int i)
{
    if (i > 0) {
        int a = 0;
    }
}

int main()
{
    int a=1, b=2;
    for (int i=0; i<4; i++) {
        b = 10;
    }
    return 0;
}
```

Variable Scope: Blocks

- We create a scope whenever we make a "block" of code using { }
- We do this all the time when we use if-else, for, while, function, etc.

```
return_type function_name(…)
{
    …
    return expression; // if any
}
```

```
if (…) {
    …
}
else {
    …
}
```

```
while (…) {
    …
}
```

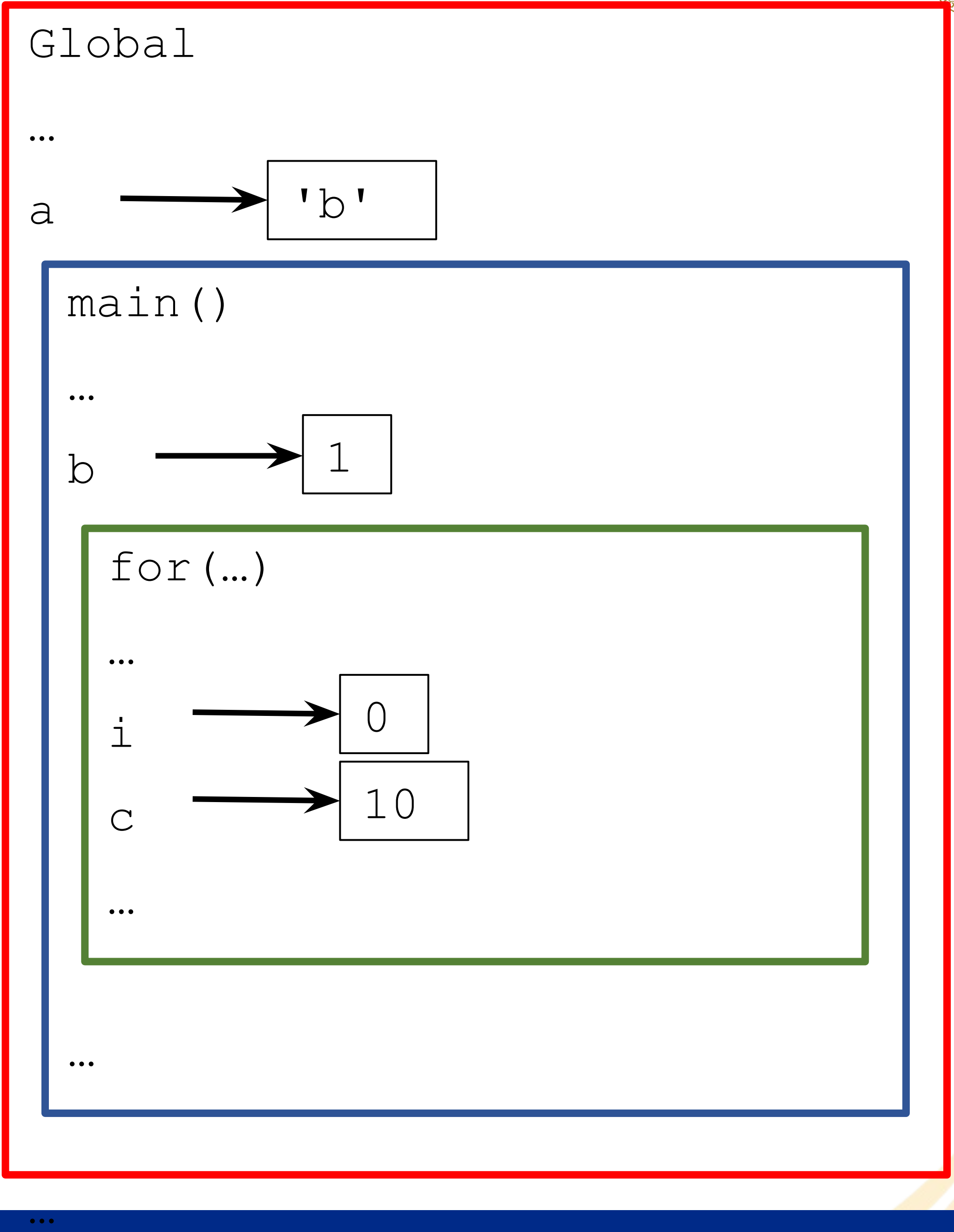
```
for (…;…;…) {
    …
}
```

Example

```
#include <stdio.h>

char a = 'b';

int main()
{
    int b=1;
    for (int i=0; i<4; i++) {
        int c=10;
        a++;
    }
    printf("%d %d", i, c);
    return 0;
}
```



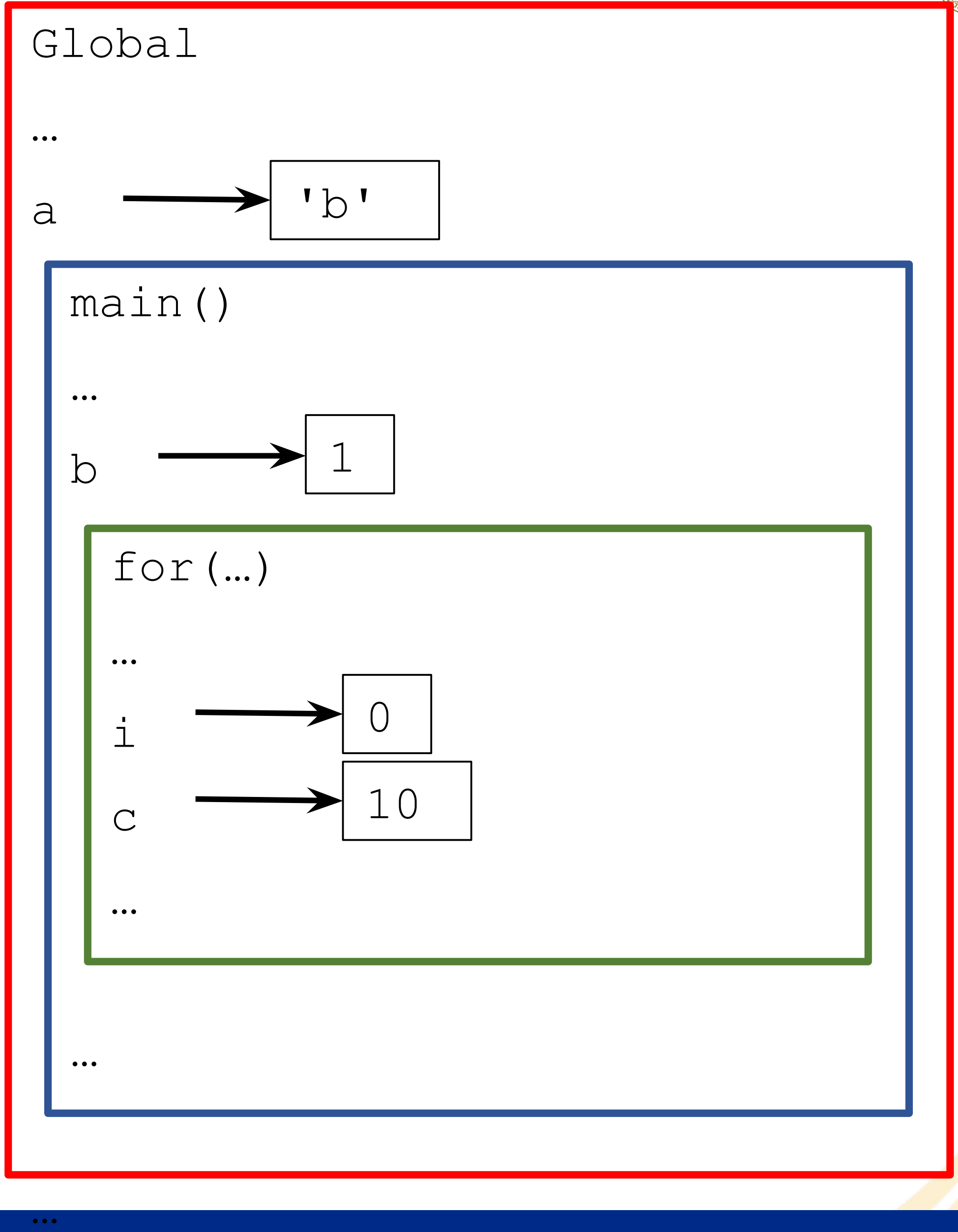
Example

```
#include <stdio.h>

char a = 'b';

int main()
{
    int b=1;
    for (int i=0; i<4; i++) {
        int c=10;
        a++;
    }
    printf("%d %d", i, c);
    return 0;
}
```

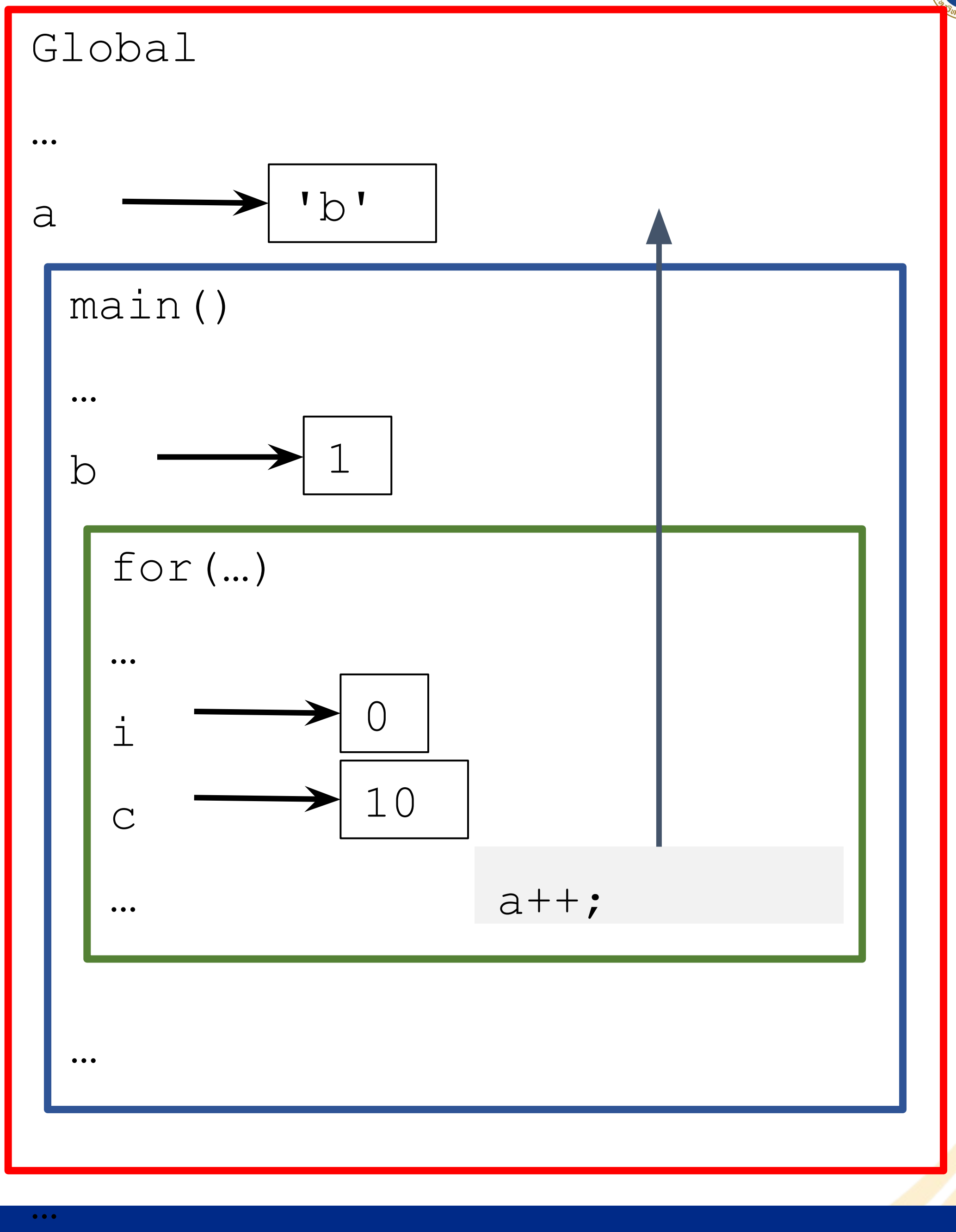
Error !!!



Example

- Variables are available after its declaration, and disappear after its scope

- General rule when looking for variables:
    - Go back until you find one
    - But you CANNOT get into any scope



Variable Scope: Override

- If variables have the same name, the most local one will be used
- For example, a variable declared within a function will override the global variable of the same name.

```c
#include<stdio.h>

char a = 'a';

int main(){
    int a;
    a = 10;
    printf("%d", a);
    return 0;
}
```

Variable Scope: Override

- If variables have the same name, the most local one will be used
- For example, a variable declared within a function will override the global variable of the same name.

```c
#include<stdio.h>

char a = 'a';

int main(){
    int a;
    a = 10;
    printf("%d", a);
    return 0;
}
```

Output

```
10
```

Variable Scope: Summary

- Variables exist within the scope of its declaration
    - They are discarded after its scope

- If there are variables with the same name, the most local one will be used (i.e., override).

- A function has its own scope, and so do other flow controls (`if-else`, `for`, `while`, …)
    - E.g., local variables of a function, variables declared in `if-else` statement, variables in `for` statement, etc.

Mahidol University

Guideline on Declaring Variables

Guideline on Declaring Variables

- Avoid making a global variable if possible (and it is almost always possible)
  - They can be modified by any statements in the file

- Avoid having variable names with the same name as the function names

- Put variable declaration early in a function

Exercise

What is the color of

- `a` at point `(A)`
- `a` at point `(B)`
- `x` at point `(B)`
- `b` at point `(B)`
- `a` at point `(C)`
- `x` at point `(C)`

{yellow, green, blue}?

Yellow - global
Green - main()
Blue - for loop

```
/* #1 */
#include <stdio.h>

char a = 'a';
float b = 1.0;
int x = 9;

int main() {
    int a = 1;
    for(int i=0; i<4; i++) {
        x = 10;
        a++;           // (A)
        b = a + x;     // (B)
    }
    x = a;             // (C)
    return 0;
}
```

Exercise

What is the output?

```
/* #2 */
#include <stdio.h>

int x = 1;

int main(){
    printf("%d %d", x, y);
    return 0;
}

int y = 2;
```

Exercise

What is the output?

```
/* #3 */
#include <stdio.h>

int x = 1;

int foo(){
   x++; // x = x + 1;
   return x + 10;
}

int main(){
    int y = foo();
    int z = foo();
    printf("%d %d %d", x, y, z);
    return 0;
}
```

Exercise

What is the value of

- `m`
- `n`
- `o`

at point `(A)` ?

```
/* #4 */
#include <stdio.h>

char m = 'c';
float n = 1.0;
int o = 9;

int main() {
    int i = 20;
    int m = 1;
    for (int i=0; i<4; i++) {
        o = 10;
        m++;
        m = 20;
        n = m + o;
    }
    o = m + i;
    // (A)
    printf("END\n");
    return 0;
}
```

Mahidol University

Lab Exercises