CSE 220 – C Programming

Program Organization

Organization

- Variables
  - Local
  - External
  - In blocks
- Scope rules:
  - Where is a variable visible
  - Lifetime: period for which the variable exists

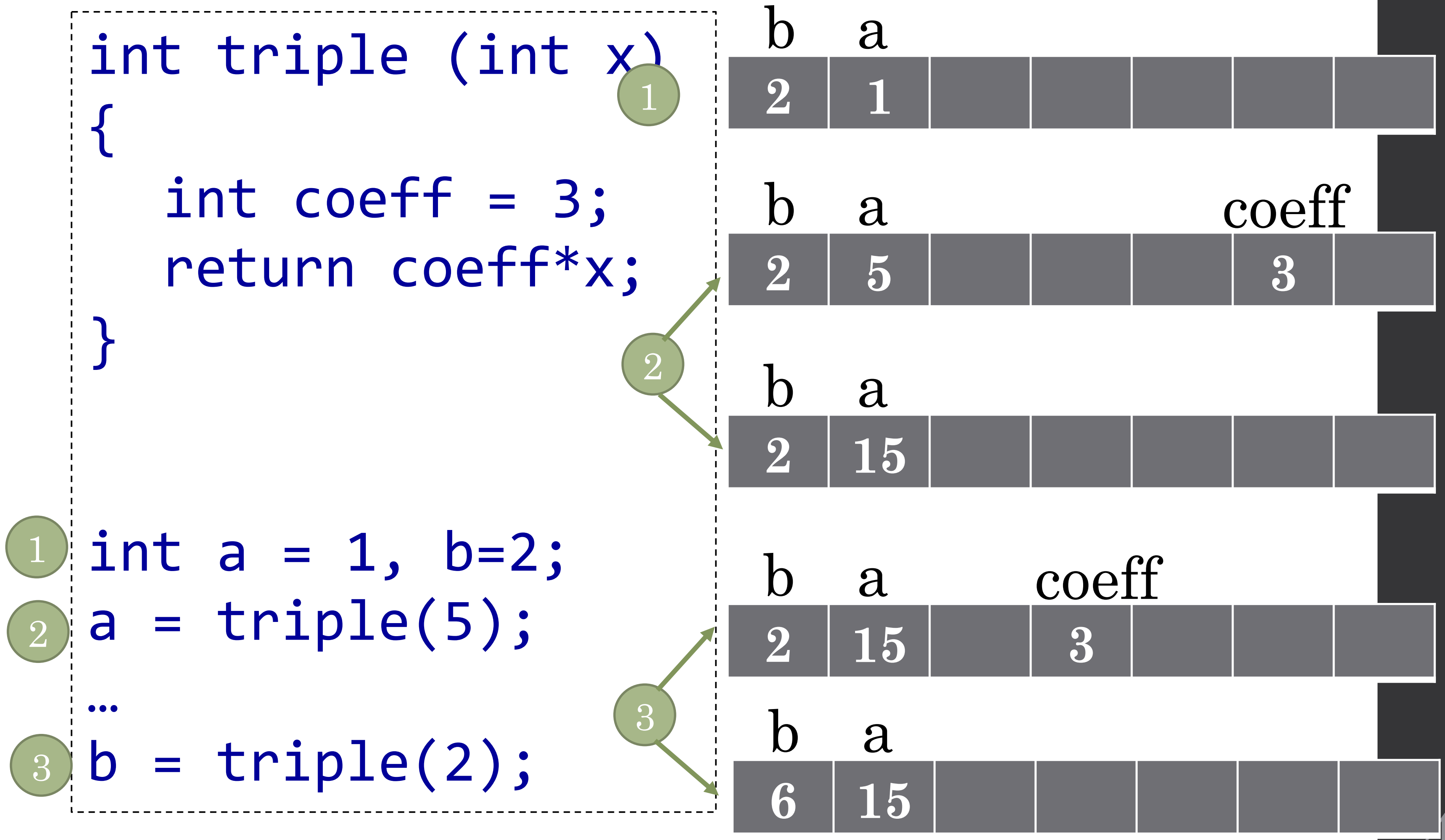
Local Variables

- A variable declared in the body of a function: <u>local</u> to the function:
  - Automatic storage duration
    - Allocated when function is called
    - Deallocated when function returns
  - Scope: Visible inside the enclosing block only

Local Variables

```
int triple (int x)
{
   int coeff = 3;
   return coeff*x;
}


int a = 1, b=2;
a = triple(5);
…
b = triple(2);
```



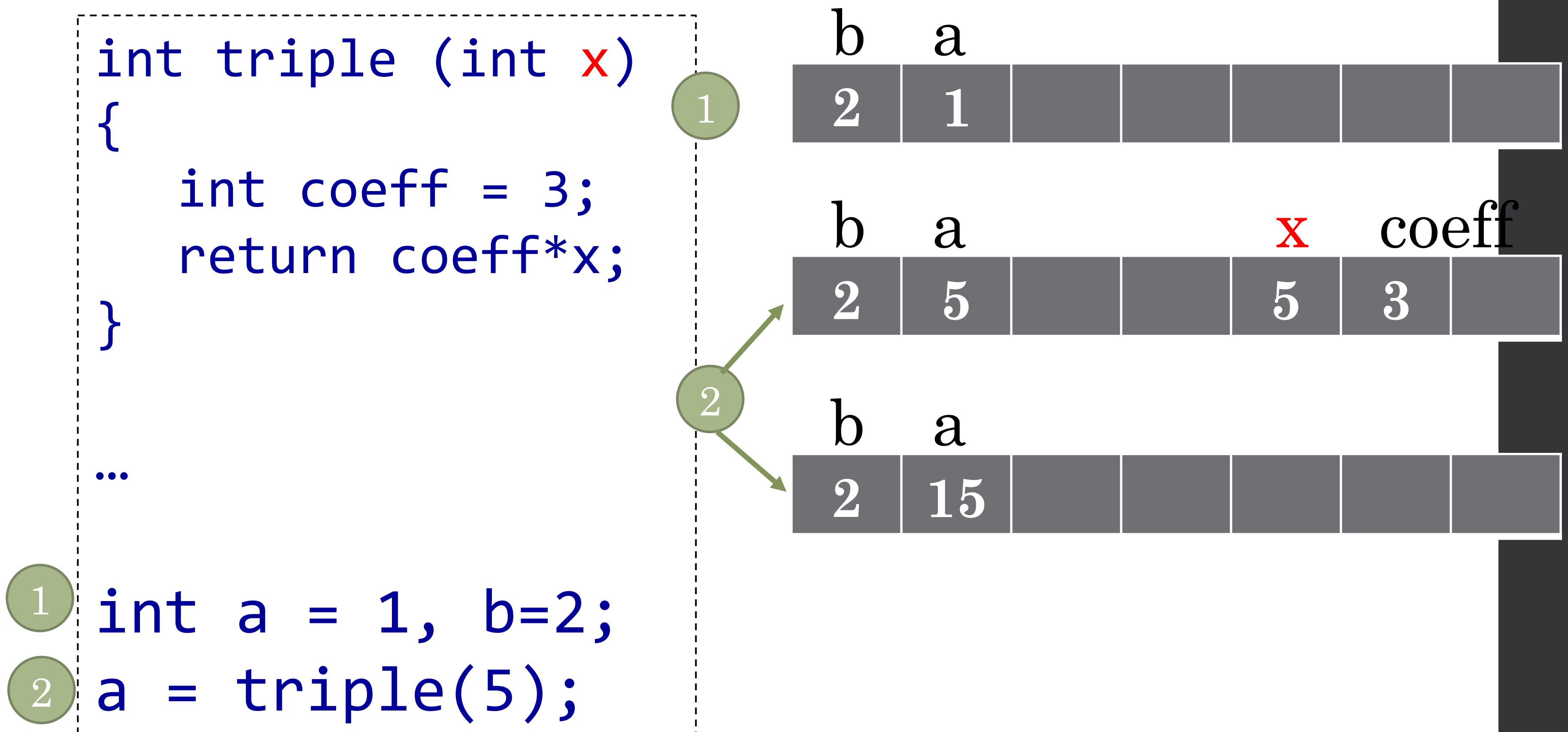
Parameters

- Similar to local variables

```
int triple (int x)
{
   int coeff = 3;
   return coeff*x;
}


…


int a = 1, b=2;
a = triple(5);
```



Static Local Variables

- Local variables declared with the <u>static</u> keyword:
  - Permanent storage duration: does not lose value
  - Occupies same memory location throughout
  - Only visible inside function

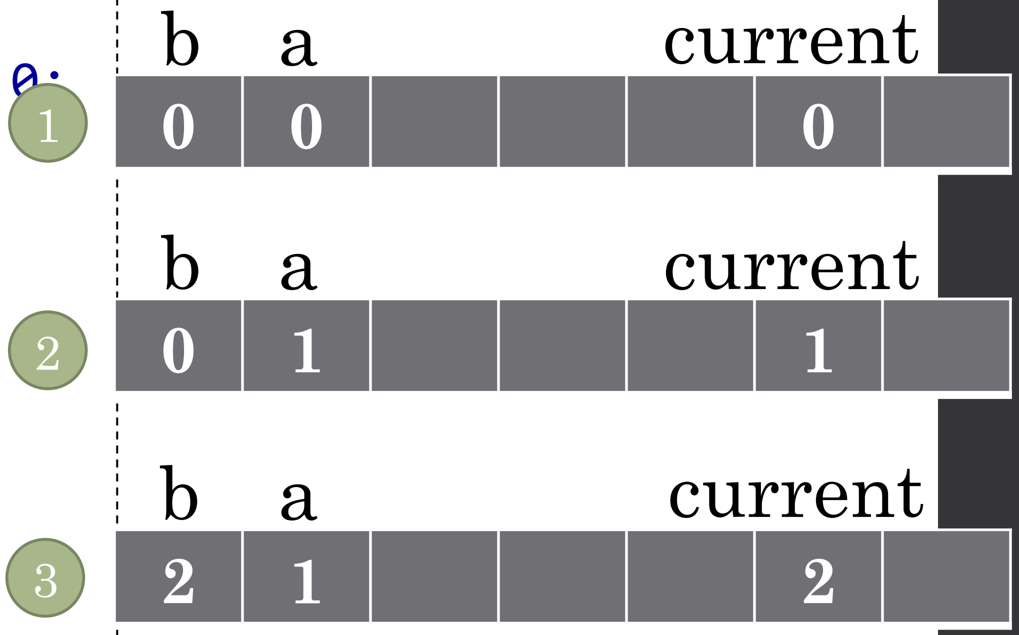
Static Variables

```
int nextNumber () {
    static int current = 0;
    current++;
    return current;
}


…
int a = 0, b=0;
a = nextNumber();
…
b = nextNumber();

printf("%d", current);
```

1 2 3



Compilation Error

External Variables

- Variables declared outside the body of a function
  - External/Global variables
  - Static storage duration
  - **File scope**: visible from declaration until end of enclosing file

External Variables

```
#include <stdio.h>
#define MAXSZ 100

int content[MAXSZ];
int last = 0;

void addOne(int x) {
     last++;
     content[last] = x;
}
```

```
int isFull() {
   return last == MAXSZ - 1;
}

int main() {
    int count;
    scanf(“%d”, &count);
    for (int i=0, i<count; i++) {
        if (!isFull()) {
             addOne(rand()%50);
        } else {
             break;
        }
}
```

External Variables

```
#include <stdio.h>
#define MAXSZ 100

int content[MAXSZ];
int last = 0;

void addOne(int x) {
     last++;
     content[last] = x;
}
```

```
int isFull() {
   return last == MAXSZ - 1;
}

int main() {
    int count;
    scanf("%d", &count);
    for (int i=0, i<count; i++) {
        if (!isFull()) {
            addOne(rand()%50);
        } else {
            break;
        }
}
```

Pros and Cons

- Convenient way for functions to share variables
- Maintenance: If type changes, we need to check every function that uses it
- If assigned wrong value: may be difficult to locate where
- Functions that rely on externals are hard to reuse

Block Variables

- Block: a compound statement

```
{
   statements
}
```

- A block variable has automatic duration

temp is created

```
if (i > j) {
        int temp = i;
        i = j;
        j = i;
}
```

temp is destroyed

Scope

- Scope: the context in which a variable is defined:
  - Duration
  - Visibility
- Scope rules: used for name resolution

Scope Rules

```
int a; /* decl 1 */
void f(int a) {    /* decl 2 */
     a = 1;
}
void g(void) {
     int a = 2;      /* decl 3 */
     if (a > 0) {
        int a;           /* decl 4 */
        a = 3;
     }
     a = 4;
}
```

```
void h(void) {
     a = 5;
}
```

- Parameter a (decl. 2) in f: hides external variable a
- Local variable a (decl. 3) hides external variable a
- Block variable a (decl. 4) hides local variable a in g
- External variable a visible in h

Program Organization (structure of a .c file)

- Preprocessing directives: #include, #define
- Type definitions: typedef (optional content)
- Declaration of external variables
- Function prototypes (declarations)
- Definition of main
- Definition of other functions

```
int a = 1;
void f(void) {
   int a = 2;
   printf(“%d”, a);
}

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

What is the output?

1. 2
2. 1
3. 0
4. Error

The declaration of local variable a inside f hides the external variable a. The local variable is printed

```
int a = 1;
void f(void) {
   int b = a;
   int a = 2;
   printf("%d", b);
}

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

What is the output?

1. 2
2. 1
3. 0
4. Error

The declaration of local variable a inside f hides the external variable a, starting from the time a was declared. Before a was declared, when b was initialized, only the external a was visible

```
int main(void) {
    {
        int x = 20;
    }
    printf(“%d”, x);
    return 0;
}
```

What is the output?

```
1. 2
2. 1
3. 0
4. Error
```

The program will not compile.
Variable x is not accessible inside the print statement.

What is the output?

```
int main(void) {
    int x = 10, y = 20;
    {
        printf(“x = %d, y = %d\n”, x, y);        x = 10, y = 20
        {
            int y = 40;
            x++;
            y++;
            printf(“x = %d, y = %d\n”, x, y);    x = 11, y = 41
        }
        printf(“x = %d, y = %d\n”, x, y);        x = 11, y = 20
    }
    return 0;
}
```