計算機程式設計

Computer Programming

Variables

Instructor: 林英嘉

2024/12/16

GitHub repo

Lifetime of C Variables

Outline

- Local variable (區域變數)
- Global variable (全域變數)
- Static variable (靜態變數)

[Illustration] Local vs. Global variables

C-course-materials/compare_local_global.c

Global variable

Local variable

```c
#include <stdio.h>
int globalVar = 100;

void do_print(void){
    printf("Global variable: %d\n", globalVar);
    // printf("Local variable: %d\n", localVar); // will cause an error
}

int main(void){
    int localVar = 0;
    printf("Global variable: %d\n", globalVar);
    do_print();
}
```

[Definition] Global variable

- The declaration of a global variable is outside any function in a program.
- In this way, all functions or code blocks in a program can use the global variable.

Scope of a Global Variable

C-course-materials/compare_local_global.c

Scope:
the whole
program

```c
#include <stdio.h>
int globalVar = 100;

void do_print(void){
    printf("Global variable: %d\n", globalVar);
    // printf("Local variable: %d\n", localVar); // will cause an error
}

int main(void){
    int localVar = 0;
    printf("Global variable: %d\n", globalVar);
    do_print();
}
```

[Definition] Local Variable

- Declaring a variable inside a function definition (including the main function) makes the variable name **local** to the code block.
- Life of a local variable:
  - Each variable's storage exists only from the declaration to the end of the block
  - Execution of the declaration allocates the storage, computes the initial value, and stores it in the variable. The end of the block deallocates the storage.

Scope of a Local Variable (1)

C-course-materials/compare_local_global.c

Scope: within the function

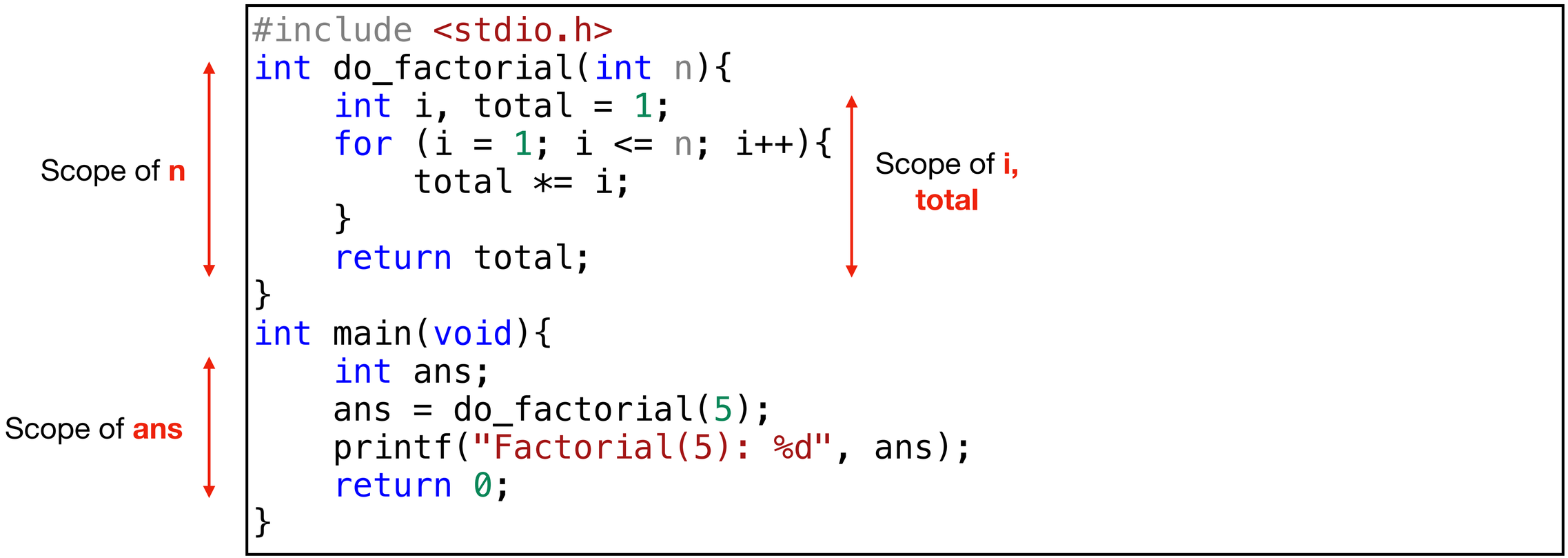
```c
#include <stdio.h>
int globalVar = 100;

void do_print(void){
    printf("Global variable: %d\n", globalVar);
    // printf("Local variable: %d\n", localVar); // will cause an error
}

int main(void){
    int localVar = 0;
    printf("Global variable: %d\n", globalVar);
    do_print();
}
```

Scope of a Local Variable (2)

C-course-materials/local_var_scope.c




```c
#include <stdio.h>
int do_factorial(int n){
    int i, total = 1;
    for (i = 1; i <= n; i++){
        total *= i;
    }
    return total;
}
int main(void){
    int ans;
    ans = do_factorial(5);
    printf("Factorial(5): %d", ans);
    return 0;
}
```

[Definition] static variable

The **static** keyword can control:

1. **Life cycle of a variable**
   - A `static` variable declared inside a function retains its value across function calls and remains in memory until the program ends.
2. **External Linkage**
   - A `static` **global** variable or function is accessible only within the file where it is declared (often called “file scope”).
   - (We will not go into detail on this today.)

Example to use a static local variable

C-course-materials/05-Functions/static_local.c

`sum` is a static local variable.

```c
int add(int a, int b){
    static int sum = 0;
    sum += (a + b);
    return sum;
}
int main(void){
    int num_a = 5;
    int num_b = 6;
    int result;
    printf("Sum:");
    for (int i = 0; i < 5; i++){
        result = add(num_a, num_b);
        printf(" %d", result);
    }
}
```

Sum: 11 22 33 44 55

`sum` is a local variable.

```c
int add(int a, int b){
    int sum = 0;
    sum += (a + b);
    return sum;
}
int main(void){
    int num_a = 5;
    int num_b = 6;
    int result;
    printf("Sum:");
    for (int i = 0; i < 5; i++){
        result = add(num_a, num_b);
        printf(" %d", result);
    }
}
```

Sum: 11 11 11 11 11