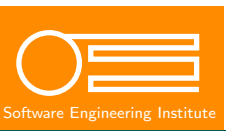
Introduction to C Programming Language
Recursive Function and Pointer
Lecture 09

Min Zhang

zhangmin@sei.ecnu.edu.cn

2020.11.30



10:00-11:40, Monday, Room 319
Software Engineering Institute, East China Normal University

Today's content

☞ **Recursive function & Pointer**

Recursive function

A recursive function is a function **that calls itself** in its definition.

Example (factorial number)

$fac(n) = 1 \times 2 \times 3 \times \ldots \times (n-1) \times n$
$fac(n) = n \times fac(n-1)$

Factorial numbers

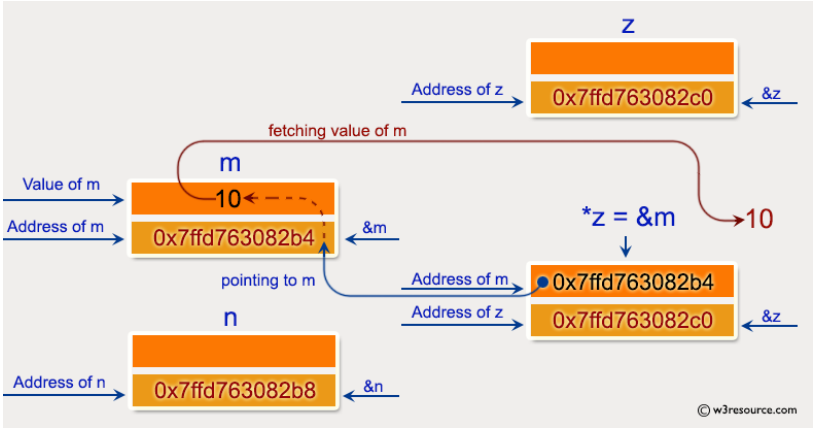
```
1  long int fac(int n){
2    if(n==0||n==1){
3      return 1;
4    }
5    return n*fac(n-1);
6  }
7
8  int main(){
9    printf("%ld",fac(10));
10 }
```

The execution trace:

`fac(10)` $\Rightarrow$
`return n*fac(9)` $\Rightarrow$
`return 10*(return 9*fac(8))` $\Rightarrow$ ... $\Rightarrow$
`return 10*(return ...(return 2*fac(1)))`.

Today's content

☞ Pointer



Recall the swap function

```c
#include <stdio.h>
void swap(int i, int j){
  int tmp=j;
  j=i;
  i=tmp;
}
int main(){
  int a=2,b=3;
  swap(a,b);
  printf("%d,%d",a,b);
  return 0;
 }
```

swap does not swap the values of a and b because when swap is called, only the values of a and b are assigned to i and j, respectively.

We want i and j represent a and b!

How two variables share the same memory?

Remember that when a variable a is declared, some bytes are allocated to it.

Memory								Address	Code
0	0	0	0	0	0	0	1	0x0000000000000000	`short int a;`
0	0	0	0	0	0	0	0	0x0000000000000001	`a=1;`
0	0	0	0	0	0	0	0	0x0000000000000002	`int *b;`
0	0	0	0	0	0	0	0	0x0000000000000003	`b=&a;`
0	0	0	0	0	0	0	0	0x0000000000000004	
0	0	0	0	0	0	0	0	0x0000000000000005	
0	0	0	0	0	0	0	0	0x0000000000000006	
0	0	0	0	0	0	0	0	0x0000000000000007	
0	0	0	0	0	0	0	0	0x0000000000000008	
0	0	0	0	0	0	0	0	0x0000000000000009	

About b here:

- b is a variable.
- The type of b is `int *`
- The value of b is 0x0000000000000000

Definition (Pointer)

In computer science, a pointer is a programming language object, whose value **refers to** (or "points to") another value stored elsewhere in the computer memory using its **memory address**.

A pointer **references** a location in memory, and obtaining the value stored at that location is known as **dereferencing** the pointer.

As an analogy, a page number in a book's index could be considered a pointer to the corresponding page; dereferencing such a pointer would be done by flipping to the page with the given page number and reading the text found on the indexed page.

Definition (Pointer)

A pointer is a variable, whose value is a memory address of some variable.

Definition (Declaration of a pointer)

`Type * pointerName;`

```
1 int *p1; // points to a place storing an integer
2 char *p2; // points to a place storing a character
3 float *p3; // points to a place storing a float number
4 double *p4; // points to a place storing a float number
```

Operations related to pointer: & and *

- &:
  - usage: `&Variable`
  - result: returns the address of the variable
  - example: `int *p=&a;`
- *:
  - usage: `*Pointer`
  - result: returns the value stored in the place where the pointer points.
  - example: `int b=*p;`

Are they equal?

```
1 int a;
2 int *p=&a;
3 *p==a
4 p==&a
```

Print out pointer using printf

```
1 int a=5;
2 int *p=&a;
3 printf("%p",p);
4 printf("%d",*p);
```

A solution to swap function

```c
void swap(int *,int *); // if you declare a function

void swap(int *i, int *j){ // if you define a function
  int tmp=*j;
  *j=*i;
  *i=tmp;
}
```

Example of calling swap function:

```c
int a=2;
int b=3;
swap(&a,&b);
```

Remember that **if the argument of a function is a pointer, it has to be given an address when the function is called.**

Recall the functions whose arguments are pointers

```
scanf("%d",&a);
swap(&a,&b); // int a,b;
strlen(str); // char str[100];
gets(str); // char str[100];
strcmp(str1,str2); // char str1[100], str2[100];
```

Array name is essentially a pointer, to continue...