

本节内容

栈的应用

——递归

王道考研/CSKAOYAN.COM

1

函数调用背后的过程

```
void main() {  
    int a, b, c;  
    ...  
    func1 (a, b);  
#1 → c=a+b;  
    ...  
}
```

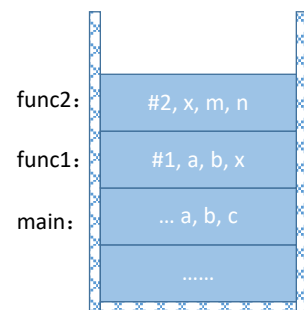
```
void func1 (int a, int b) {  
    int x;  
    ...  
    func2 (x);  
#2 → x=x+10086;  
    ...  
}
```

```
void func2 (int x) {  
    int m, n;  
    ...  
}
```

函数调用的特点：最后被调用的函数最先执行结束（LIFO）

函数调用时，需要用一个栈存储：

- ① 调用返回地址
- ② 实参
- ③ 局部变量



函数调用栈

王道考研/CSKAOYAN.COM

2

函数调用背后的过程

The screenshot shows a C++ IDE with the following code:

```

170 int func2 (int x) {
171     int m, n;
172     m = x + 1;
173     n = x + 2;
174 }
175
176 int func1 (int a, int b) { a: 1 b: 2
177     int x= a+b; x: 3
178     func2 (x);
179     x = x+10086;
180 }
181
182 int main() {
183     int a = 1, b = 2, c = 3;
184     func1(a, b);
185     c = a+b;
186 }

```

The debugger stack shows the following frames (from top to bottom):

- Thread-1-c...ain-thread>
- func2(int) main.cpp:173
- func1(int, int) main.cpp:178
- main main.cpp:184
- start 0x00007fff713d13d5

The variables window shows the following values:

- x = (int) 3
- m = (int) 4
- n = (int) 0

王道考研/CSKAOYAN.COM

3

栈在递归中的应用

适合用“递归”算法解决：可以把原始问题转换为属性相同，但规模较小的问题

Eg 1: 计算正整数的阶乘 $n!$

$$\text{factorial}(n) = \begin{cases} n * \text{factorial}(n-1), & n > 1 \\ 1, & n = 1 \\ 1, & n = 0 \end{cases}$$

递归表达式 (递归体)

边界条件 (递归出口)

Eg 2: 求斐波那契数列

$$\text{Fib}(n) = \begin{cases} \text{Fib}(n-1) + \text{Fib}(n-2), & n > 1 \\ 1, & n = 1 \\ 0, & n = 0 \end{cases}$$

王道考研/CSKAOYAN.COM

4

栈在递归中的应用

Eg 1: 递归算法求阶乘

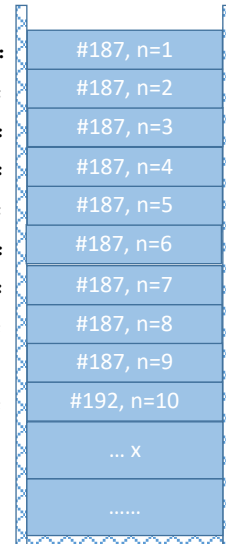
```

182 //计算正整数 n!
183 int factorial (int n){
184     if (n==0 || n==1)
185         return 1;
186     else
187         return n*factorial(n-1);
188 }
189
190 int main() {
191     //... 其他代码
192     int x=factorial(10);
193     printf("奥利给! ");
194 }

```

再次思考：递归算法的空间复杂度

(第10层):
 (第9层):
 (第8层):
 (第7层):
 (第6层):
 (第5层):
 (第4层):
 (第3层):
 递归函数factorial (第2层):
 递归函数factorial (第1层):
 main:



函数调用栈

递归调用时，函数调用栈可称为“递归工作栈”
 每进入一层递归，就将递归调用所需信息压入栈顶
 每退出一层递归，就从栈顶弹出相应信息

缺点：太多层递归可能会导致栈溢出

王道考研/CSKAOYAN.COM

5

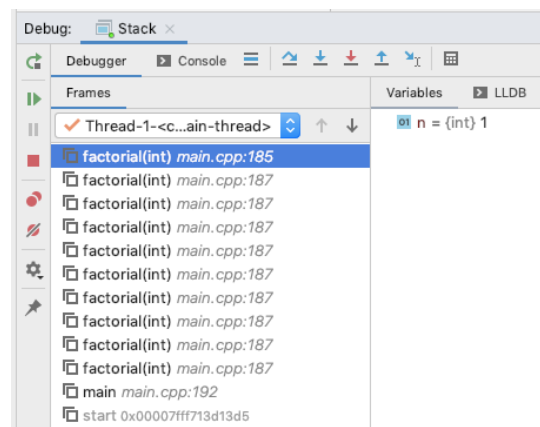
栈在递归中的应用

Eg 1: 递归算法求阶乘

```

182 //计算正整数 n!
183 int factorial (int n){
184     if (n==0 || n==1)
185         return 1;
186     else
187         return n*factorial(n-1);
188 }
189
190 int main() {
191     //... 其他代码
192     int x=factorial(10);
193     printf("奥利给! ");
194 }

```



可以自定义栈将递归算法改造成非递归算法

王道考研/CSKAOYAN.COM

6

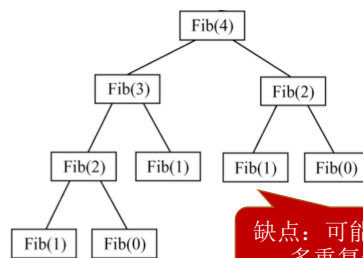
栈在递归中的应用

Eg 2: 递归算法求斐波那契数列

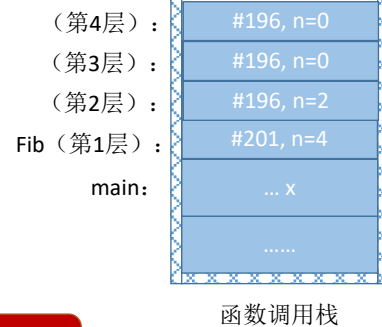
```

190 int Fib(int n){
191     if(n==0)
192         return 0;
193     else if(n==1)
194         return 1;
195     else
196         return Fib(n-1)+Fib(n-2);
197 }
198
199 int main() {
200     //... 其他代码
201     int x=Fib(4);
202     printf("奥利给! ");
203 }

```



缺点：可能包含很多重复计算



王道考研/CSKAOYAN.COM

7

知识回顾与重要考点

函数调用的特点：最后被调用的函数最先执行结束（LIFO）

函数调用时，需要用“函数调用栈”存储：

- ① 调用返回地址
- ② 实参
- ③ 局部变量

递归调用时，函数调用栈可称为“递归工作栈”
每进入一层递归，就将递归调用所需信息压入栈顶
每退出一层递归，就从栈顶弹出相应信息

缺点：效率低，太多层递归可能会导致栈溢出；可能包含很多重复计算

可以自定义栈将递归算法改造成非递归算法

王道考研/CSKAOYAN.COM

8



@王道论坛



等撩

@王道计算机考研备考
@王道咸鱼老师-计算机考研
@王道楼楼老师-计算机考研



等撩



@王道计算机考研



@王道计算机考研



微信视频号

@王道计算机考研



微信公众平台

@王道在线