C 错误处理 C 可变参数

C 递归

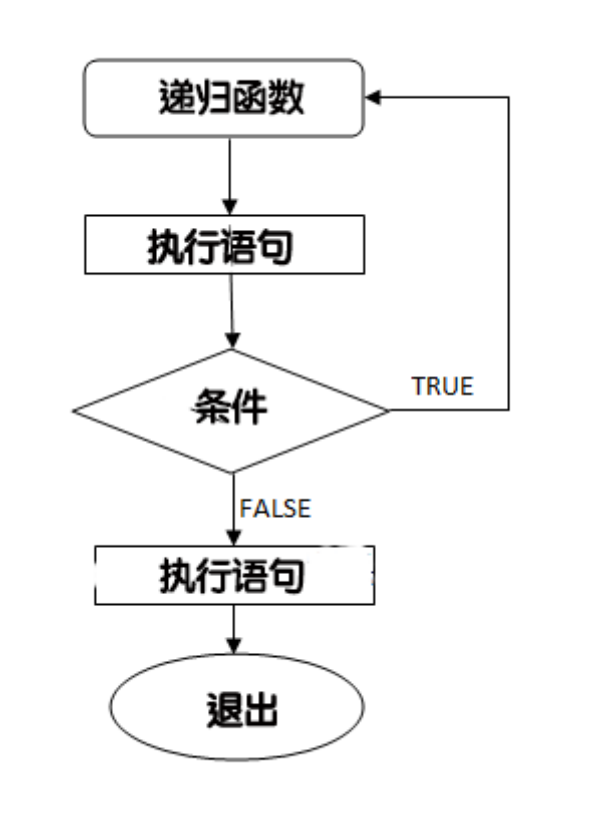
递归指的是在函数的定义中使用函数自身的方法。

> *举个例子：*
> *从前有座山，山里有座庙，庙里有个老和尚，正在给小和尚讲故事呢！故事是什么呢？"从前有座山，山里有座庙，庙里有个老和尚，正在给小和尚讲故事呢！故事是什么呢？'从前有座山，山里有座庙，庙里有个老和尚，正在给小和尚讲故事呢！故事是什么呢？……'"*

语法格式如下：

```
void recursion()
{
statements;
... ... ...
recursion(); /* 函数调用自身 */
... ... ...
}
int main()
{
recursion();
}
```

流程图：



C 语言支持递归，即一个函数可以调用其自身。但在使用递归时，程序员需要注意定义一个从函数退出的条件，否则会进入死循环。

递归函数在解决许多数学问题上起了至关重要的作用，比如计算一个数的阶乘、生成斐波那契数列，等等。

数的阶乘

下面的实例使用递归函数计算一个给定的数的阶乘：

实例

```
#include <stdio.h>
double factorial(unsigned int i)
{
if(i <= 1)
{
return 1;
}
return i * factorial(i - 1);
}
int main()
{
int i = 15;
printf("%d 的阶乘为 %f\n", i, factorial(i));
return 0;
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
15 的阶乘为 1307674368000.000000
```

斐波那契数列

下面的实例使用递归函数生成一个给定的数的斐波那契数列：

实例

```
#include <stdio.h>
int fibonaci(int i)
{
if(i == 0)
{
return 0;
}
if(i == 1)
{
return 1;
}
return fibonaci(i-1) + fibonaci(i-2);
}
int main()
{
int i;
```

```
for (i = 0; i < 10; i++)
{
printf("%d\t\n", fibonaci(i));
}
return 0;
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
0
1
1
2
3
5
8
13
21
34
```

C 错误处理 C 可变参数

2 篇笔记

写笔记

递归是一个简洁的概念，同时也是一种很有用的手段。但是，使用递归是要付出代价的。与直接的语句(如while循环)相比，递归函数会耗费更多的运行时间，并且要占用大量的栈空间。递归函数每次调用自身时，都需要把它的状态存到栈中，以便在它调用完自身后，程序可以返回到它原来的状态。未经精心设计的递归函数总是会带来麻烦。

风中追柳 2年前 (2017-05-23)

采用递归方法来解决问题，必须符合以下三个条件：

1、可以把要解决的问题转化为一个新问题，而这个新的问题的解决方法仍与原来的解决方法相同，只是所处理的对象有规律地递增或递减。

说明：解决问题的方法相同，调用函数的参数每次不同（有规律的递增或递减），如果没有规律也就不能适用递归调用。

2、可以应用这个转化过程使问题得到解决。

说明：使用其他的办法比较麻烦或很难解决，而使用递归的方法可以很好地解决问题。

3、必定要有一个明确的结束递归的条件。

说明：一定要能够在适当的地方结束递归调用。不然可能导致系统崩溃。

风中的海洋 2年前 (2017-07-26)