C 递归 C 内存管理

C 可变参数

有时，您可能会碰到这样的情况，您希望函数带有可变数量的参数，而不是预定义数量的参数。C 语言为这种情况提供了一个解决方案，它允许您定义一个函数，能根据具体的需求接受可变数量的参数。下面的实例演示了这种函数的定义。

```
int func(int, ... )
{
.
.
.
}
int main()
{
func(2, 2, 3);
func(3, 2, 3, 4);
}
```

请注意，函数 **func()** 最后一个参数写成省略号，即三个点号（**...**），省略号之前的那个参数是 **int**，代表了要传递的可变参数的总数。为了使用这个功能，您需要使用 **stdarg.h** 头文件，该文件提供了实现可变参数功能的函数和宏。具体步骤如下：

- 定义一个函数，最后一个参数为省略号，省略号前面可以设置自定义参数。
- 在函数定义中创建一个 **va_list** 类型变量，该类型是在 stdarg.h 头文件中定义的。
- 使用 **int** 参数和 **va_start** 宏来初始化 **va_list** 变量为一个参数列表。宏 va_start 是在 stdarg.h 头文件中定义的。
- 使用 **va_arg** 宏和 **va_list** 变量来访问参数列表中的每个项。
- 使用宏 **va_end** 来清理赋予 **va_list** 变量的内存。

现在让我们按照上面的步骤，来编写一个带有可变数量参数的函数，并返回它们的平均值：

实例

```
#include <stdio.h>
#include <stdarg.h>
double average(int num,...)
{
va_list valist;
double sum = 0.0;
int i;
/* 为 num 个参数初始化 valist */
va_start(valist, num);
/* 访问所有赋给 valist 的参数 */
for (i = 0; i < num; i++)
{
sum += va_arg(valist, int);
}
/* 清理为 valist 保留的内存 */
va_end(valist);
return sum/num;
```

```
}
int main()
{
printf("Average of 2, 3, 4, 5 = %f\n", average(4, 2,3,4,5));
printf("Average of 5, 10, 15 = %f\n", average(3, 5,10,15));
}
```

当上面的代码被编译和执行时，它会产生下列结果。应该指出的是，函数 **average()** 被调用两次，每次第一个参数都是表示被传的可变参数的总数。省略号被用来传递可变数量的参数。

```
Average of 2, 3, 4, 5 = 3.500000
Average of 5, 10, 15 = 10.000000
```

← C 递归

C 内存管理 →