Compilation of **Common Custom Functions** in C Language

by Jaser Li (HDU) Sept. 2023

【编者注：该部分内容非常重要，值得牢记甚至背下来！】

* 十进制转二进制

void **decimalToBinary**(int *num*)

{

    if(*num*==0)

    {

**printf**("0");

        return; // 思考：此处return的作用是？

    }

    int binary[32];

    int i = 0;

    while(*num*>0)

    {

        binary[i]=*num*%2;

*num*=*num*/2;

        i++;

    }

    for(int j=i-1;j>=0;j--) // 逆序输出

**printf**("%d",binary[j]);

}

【拓展】该代码段中"return"的作用

在这个代码示例中，return的作用是在函数decimalToBinary中使用，用于提前结束函数的执行并返回到调用该函数的地方。

在decimalToBinary函数中，当输入的十进制数为0时，会直接输出字符串"0"并立即返回，不再执行后续的代码。这是因为0的二进制表示就是0，不需要执行其他转换操作。因此，通过在这种情况下使用return语句，可以避免执行多余的代码，并确保只输出"0"一次。

而在主函数main中，调用decimalToBinary函数后，在输出二进制数之后，也使用了return 0;来结束程序的执行并返回0作为退出状态码。

return语句在函数中的作用是返回一个值给调用者。例如，decimalToBinary函数的返回类型是void，即无返回值，因此return语句只是用于提前结束函数的执行。

* 十进制转十六进制

void **decimalToHexadecimal**(int *num*)

{

    if(*num*==0)

    {

**printf**("0");

        return;

    }

    char hex[32];

    int i=0;

    while(*num*>0)

    {

        int remainder=*num*%16; // remainder存放余数

        if(remainder<10)

            hex[i]='0'+remainder; // 将余数转换为对应的字符

        else

            hex[i]='A'+(remainder-10); // 将余数转换为对应的字符

*num*/=16;

        i++;

    }

    for(int j=i-1;j>=0;j--)

**printf**("%c",hex[j]);

}

* 阶乘函数【递归】

int **factorial**(int *num*)

{

    if(*num*==0 || *num*==1) return 1; // 规定0!=1，勿漏分类！

    else return *num*\***factorial**(*num*-1);

}

如果num等于0或1，即输入为0或1，那么函数直接返回1（因为0!和1!均为1）。

如果num不是0或1，那么函数通过调用自身来计算num的阶乘。具体步骤如下：

函数将num与num-1相乘，并将结果作为返回值。

在每次递归调用中，num的值都减1，使得问题规模逐渐减小，直到达到基本情况（即num为0或1）。

通过递归调用的方式，函数能够不断地将原始问题拆分为更小的子问题，并最终得到问题的解。在该函数中，递归调用一直执行到基本情况，然后逐层返回结果，最终得到输入整数的阶乘值。

* 求最大公约数、最小公倍数【欧几里德算法】

#include <stdio.h>

int **gcd**(int *a*,int *b*)

{

    while(*b*!=0){

        int temp=*a*; // 创建一个临时变量temp并将其赋值为a的值

*a*=*b*; // 将a的值更新为b

*b*=temp%*b*; // 将b的值更新为temp除以b的余数

    }

    return *a*;

}

int **lcm**(int *a*,int *b*)

{

    return (*a*\**b*)/**gcd**(*a*,*b*);

}

输入样例：32 72

输出样例：8 288

* 求一个整数的逆序数

函数接口定义：

int reverse(int number);//其中函数reverse须返回用户传入的整型number的逆序数。

输入样例：-12340

输出样例：-4321（去无意义零）

int **reverse**(int *number*)

{

    int reversed=0;

    int isNegative=0;

    if(*number*<0)

    {

        isNegative=1;

*number*=-*number*;

    }

    while(*number*!=0)

    {

        reversed=reversed\*10+*number*%10;

*number*/=10;

    }

    if(isNegative) return-reversed;

    return reversed;

}

函数接受一个整数参数number，并返回该整数的反转值。

函数内部定义了三个整数变量：

reversed：用于存储反转后的整数。

isNegative：标志是否为负数的变量，初始值为0。

number：在后续的操作中会被改变。

首先，函数会判断输入的number是否小于0。如果是负数，将isNegative设置为1，并将number变为其绝对值。

接下来，在一个循环中，当number不等于0时，会执行以下操作：

将reversed的值乘以10并加上number对10取余的结果，以实现反转数字的效果。

将number除以10，用于获取下一位数字。

通过反复执行上述步骤，直到number的值为0，循环结束。此时，reversed中存储的就是输入整数的反转值。

最后，如果isNegative的值为1，表示原始输入是负数，函数会返回-reversed，即反转值的负数形式；否则，函数返回reversed，即反转值的正数形式。