C语言中函数的分类

1.库函数

IO函数

字符串操作函数

字符操作函数

内存操作函数

时间/日期函数

数学函数

其他库函数

www.cplusplus.com

www.zh.cppreference.com

字符串操作函数

拷贝函数 - strcpy

#include <stdio.h>

#include <string.h>

int main()

{

    char arr1[] = "hello";

    char arr2[20] = "##############";

    strcpy(arr2,arr1);  //将arr1的内容拷贝到arr2 拷贝时也会将/0也拷贝进去

    printf("%s\n",arr2);

    return 0;

}

memset是一个初始化函数，作用是将某一块内存中的全部设置为指定的值。

void \*memset(void \*s, int c, size\_t n);

s指向要填充的内存块。

c是要被设置的值。

n是要被设置该值的字符数。

返回类型是一个指向存储区s的指针。

#include <stdio.h>

#include <string.h>

int main()

{

    char arr[] = "hello world";

    memset(arr,'\*',5);

    printf("%s\n",arr);

    //\*\*\*\*\* world

    return 0;

}

2.自定义函数

自定义函数的简单示例

#include <stdio.h>

int Add(int x, int y)   //Add前面的int 是表示这个函数最终会返回一个整型值

{

    int z = 0;

    z = x + y;

    return z;

}

int main()

{

    int a = 10;

    int b = 20;

    int sum = Add(a,b);

    printf("%d\n",sum);

    return 0;

}

找出两个数的较大值

#include <stdio.h>

#include <string.h>

int get\_max(int x, int y)

{

    if(x>y)

        return x;

    else

        return y;

}

int main()

{

    int a = 10;

    int b = 20;

    int max = get\_max(a, b);

    printf("%d\n",max);

    return 0;

}

写一个函数交换两个整形的内容

需要运用指针进行操作

#include <stdio.h>

#include <string.h>

void Swap1(int x, int y)     //void意思是这个函数没有返回值，void意思为空

{

    int tmp = 0;

    tmp = x;

    x = y;

    y = tmp;

}

//-------以上操作不能将两个整数类型进行互换--------

//当实参传给形参的时候，形参其实是实参的一份临时拷贝，对形参的修改是不会改变实参的

void Swap2(int\* pa,int\* pb)     //void意思是这个函数没有返回值，void意思为空

{

    int tmp = 0;

    tmp = \*pa;

    \*pa = \*pb;

    \*pb = tmp;

}

int main()

{

    int a = 10;

    int b = 20;

    printf("a=%d b=%d\n", a, b);

    Swap2(&a, &b);

    printf("a=%d b=%d\n", a, b);

    return 0;

}

// int main()

// {

//     int a = 0;

//     int\* pa = &a;    //pa用于存放a的地址，pa此时为指针变量，用int\*进行修饰

//     \*pa = 20;        //解引用操作

//     return 0;

// }

**实际参数与形式参数**

实际参数（实参）：

真实传给函数的参数。可以是：常量、变量、表达式、函数等。无论实参是何种类型，在进行函数调用时，他们都必须有确定的值，以便把这些值传给形参。

形式参数（形参）：

是指函数名后括号中的变量，形式参数只有在函数被调用的过程中才实例化（分配内存空间），当函数调用结束之后自动销毁。

**形参和实参使用的不是同一空间**

**函数的调用**

传值调用

传址调用

使用函数判断素数

#include <stdio.h>

#include <math.h>

int is\_prime(int x)

{

    int y = 0;

    for (y=2;y<=sqrt(x);y++)

    {

        if (x%y==0)

            return 0;

    }

    return 1;

}

int main()

{

    int i = 0;

    for (i = 101;i<=200;i+=2)

    {

        if (is\_prime(i) == 1)

            printf("%d ", i);

    }

    return 0;

}

使用函数判断闰年

#include <stdio.h>

int is\_leap\_year(int x)

{

    if ((x%4==0&&x%100!=0)||(x%400==0))

        return 1;

    else

        return 0;

}

int main()

{

    int year = 0;

    for (year=1000;year<2001;year++)

    {

        if(is\_leap\_year(year) == 1)

        {

            printf("%d ", year);

        }

    }

    return 0;

}

使用函数进行二分查找

#include <stdio.h>

//二分查找

                     //本质上这个arr是个指针

int binary\_search(int arr[], int k, int sz)

{

    //int sz = sizeof(arr)/sizeof(arr[0]);      //通过传参进入函数的数组，不能通过这个方法进行计算元素个数

    int left = 0;

    int right = sz-1;

    while (left<=right)

    {

        int mid = (left+right)/2;

        if(arr[mid] > k)

        {

            right = mid-1;

        }

        else if(arr[mid] < k)

        {

            left = mid+1;

        }

        else

        {

            return mid;

        }

    }

    return -1;

}

int main()

{

    int arr[] = {1,2,3,4,5,6,7,8,9};

    int k = 7;

    int sz = sizeof(arr)/sizeof(arr[0]);

                            //此时下面这个arr为数组arr首元素的地址

    int ret = binary\_search(arr, k, sz);

    if (ret == -1)

        printf("没找到");

    else

        printf("找到了，下标为:%d\n",ret);

    return 0;

}

计算函数的调用次数

#include <stdio.h>

//每一次调用Add num都会加一

void Add(int\* p)

{

    (\*p)++;

}

int main()

{

    int num = 0;

    Add(&num);

    printf("%d\n",num);

    Add(&num);

    printf("%d\n",num);

    Add(&num);

    printf("%d\n",num);

    return 0;

}

**函数的嵌套调用和链式访问**

链式访问：把一个函数的返回值作为另外一个函数的参数

#include <stdio.h>

int main()

{

    printf("%d",printf("%d",printf("%d",43)));

    //4321

    //printf - 返回值是打印在屏幕上的字符串的个数

    return 0;

}

**函数的声明和定义**

**函数的递归**

程序调用自身的编程技巧称为递归。递归的主要思想方式在于：把大事化小

#include <stdio.h>

int main()

{

    printf("world\n");

    main();

    return 0;

}

上面代码会造成 stack overflow(栈溢出)

将1234 拆分为1 2 3 4

#include <stdio.h>

void print(int n)

{

    if (n>9)

    {

        print(n/10);

    }

    printf("%d ",n%10);

}

int main()

{

    unsigned int num = 0;

    scanf("%d",&num);   //1234

    //递归

    print(num);

    return 0;

}

**递归的两个必要条件**

存在限制条件，当满足这个限制条件的时候，递归将不再继续

每次递归调用之后越来越接近这个限制条件

**不使用strlen函数计算数组的长度**

#include <stdio.h>

//不使用strlen函数计算数组的长度

// int my\_strlen(char\* str)

// {

//     //计算字符串的长度

//     int count = 0;

//     while (\*str != '\0')

//     {

//         count++;

//         str++;

//     }

//     return count;

// }

//不创建临时变量计算数组的长度

int my\_strlen(char\* str)

{

    if (\*str != '\0')

    {

        return 1+my\_strlen(str+1);

    }

    else

        return 0;

}

int main()

{

    char arr[] = "bit";

    int len = my\_strlen(arr);

    printf("len = %d\n", len);

    return 0;

}

**递归计算阶乘**

#include <stdio.h>

int Fcal(int k)

{

    if (k<=1)

        return 1;

    else

    {

        return k \* Fcal(k-1);

    }

}

// int Fcal(int k)

// {

//     int i = 0;

//     int ret = 1;

//     for (i=1;i<=k;i++)

//     {

//         ret \*= i;

//     }

//     return ret;

// }

int main()

{

    int n = 0;

    int ret = 0;

    scanf("%d",&n);

    ret = Fcal(n);

    printf("%d\n",ret);

    return 0;

}

**计算第n个波菲纳契数**

#include <stdio.h>

//计算第n个波菲纳契数，使用递归不合适

//int count = 0;

// int Fib(int i)

// {

//     if (i == 3)     //测试第三个波菲纳契数的计算次数

//     {

//         count++;

//     }

//     if(i<=2)

//         return 1;

//     else

//         return Fib(i-1) + Fib(i-2);

// }

int Fib(int n)

{

    int a = 1;

    int b = 1;

    int c = 1;

    while (n>2)

    {

        c = a + b;

        a = b;

        b = c;

        n--;

    }

    return c;

}

int main()

{

    int n = 0;

    int ret= 1;

    scanf("%d",&n);

    ret = Fib(n);

    printf("%d\n",ret);

    //printf("%d\n",count);

    return 0;

}

**汉诺塔问题**

**青蛙跳台阶问题**

**剑指offer**