STUACM 第一次集训

2019/09/28

- 简单介绍C语言函数
  - 引子

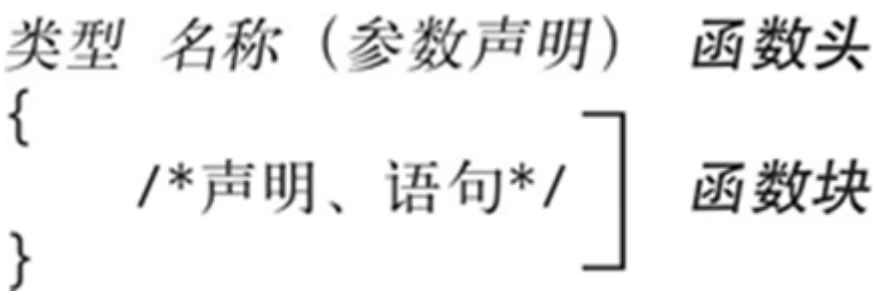
```
// 判断一个整数是否在区间[0,10)或[45,55)或[90,100)
bool is_satisfied(int n){
  return (0<=n&&n<10)||(45<=n&&n<55)||(90<=n&&n<100);
}
int main()
{
  // 假设判断a, b, c, d分别是否满足
  int a = 5, b = 10, c = 20, d = 95;

  // 不用函数封装，直接写
  bool res_a_0 = (0<=a&&a<10)||(45<=a&&a<55)||(90<=a&&a<100);
  bool res_b_0 = (0<=b&&b<10)||(45<=b&&b<55)||(90<=b&&b<100);
  bool res_c_0 = (0<=c&&c<10)||(45<=c&&c<55)||(90<=c&&c<100);
  bool res_d_0 = (0<=d&&d<10)||(45<=d&&d<55)||(90<=d&&d<100);

  // 用函数封装这一功能后
  bool res_a_1 = is_satisfied(a);
  bool res_b_1 = is_satisfied(b);
  bool res_c_1 = is_satisfied(c);
  bool res_d_1 = is_satisfied(d);

  /*
  用函数把同一功能的代码封装起来后，直接输入参数来调用函数即可得到结果，
  可以明显地提高编程速度和质量，
  函数名又能起到一定的解释作用，代码显得更加清晰易读
  */
  return 0;
}
```

  - 函数定义



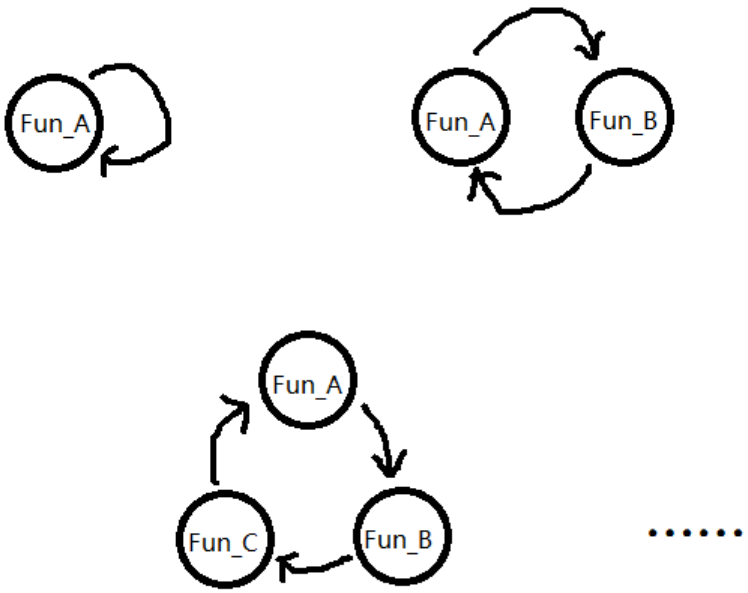
  - 例子

```
// 计算圆的面积
double cal_circle(double r){
  const int pi = 3.1415926;
  double S = pi * r * r;
  return S;
}
```

- 函数调用函数

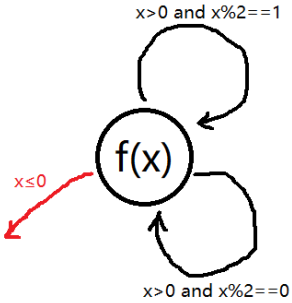
```
// 计算圆的面积
double cal_circle(double r){
  const int pi = 3.1415926;
  double S = pi * r * r;
  return S;
}
// 计算圆环的面积
double cal_area(double r1, double r2){
  if(r1>r2){
      int t = r1;
      r1 = r2;
      r2 = r;
  }
  return cal_circle(r2) - cal_circle(r1);
}
int main(){
  double S = cal_area(50,10);
  return 0;
}
```

- 递归函数



- 例子

- $f(a)=\begin{cases}0, & a\le 0\\ a+f(a-1), & a>0\ and\ a\%2==1\\ a*f\left(\frac{a}{2}\right), & a>0\ and\ a\%2==0\end{cases}$



为了防止函数陷入死循环，递归过程应该是要向某个“方向”不断变化（*该例子中a总是在减小*），**并且在该方向上累积到某个限制时跳出递归**（*该例子中a≤0便跳出递归*）。

  - 递归与非递归

```
// 递归实现
int f(int a){
  if(a<=0)
      return 0;
  if(a%2==1)
      return a+f(a-1);
  if(a%2==0)
      return a*f(a/2);
}
```

```
// 非递归实现
int f[1000] = {0};
int cal_f(int a){    // a<1000
  for(int x=1;x<=a;x++){
      if(x%2==1){
          f[x] = x + f[x-1];
      }else{
          f[x] = x * f[x/2];
      }
  }
}
```

- 最大公因数(GCD)

辗转相除法

  - 实现

```
int gcd(int a, int b){
  return b==0?a:gcd(b,a%b);
}
```

  - 例子

gcd(75, 27) = 3

a	b	a%b
75	27	21
27	21	6
21	6	3
6	3	0
3	0	

- 证明

  https://blog.csdn.net/z69183787/article/details/64126152

- 最小公倍数(LCM)
  - 实现

    ```
    int lcm(int a, int b){
        return a/gcd(a,b)*b;
    }
    ```

  - 例子

    gcd(75, 27) = 3
    lcm(75, 27) = 75 * 27 / gcd(75, 27) = 675

  - 证明

    https://blog.csdn.net/z69183787/article/details/64126152

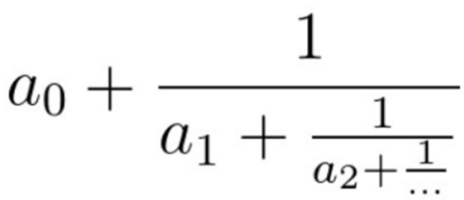
- 两道例题的讨论与分析
  - **problem 1**

LCP 2. 分式化简

难度 简单 2 收藏 分享 切换为英文

题目描述 评论 (10) 题解(6) New 提交记录

有一个同学在学习分式。他需要将一个连分数化成最简分数，你能帮助他吗？



连分数是形如上图的分式。在本题中，所有系数都是大于等于0的整数。

输入的 `cont` 代表连分数的系数（`cont[0]` 代表上图的 a_0，以此类推）。返回一个长度为2的数组 `[n, m]`，使得连分数的值等于 `n / m`，且 `n, m` 最大公约数为1。

示例 1：

```
输入：cont = [3, 2, 0, 2]
输出：[13, 4]
解释：原连分数等价于3 + (1 / (2 + (1 / (0 + 1 / 2))))。注意[26, 8], [-13, -4]都不是正确答案。
```

示例 2：

```
输入：cont = [0, 0, 3]
输出：[3, 1]
解释：如果答案是整数，令分母为1即可。
```

限制：

1. `cont[i] >= 0`
2. `1 <= cont的长度 <= 10`
3. `cont` 最后一个元素不等于0
4. 答案的 `n, m` 的取值都能被32位int整型存下（即不超过 `2 ^ 31 - 1`）。

- **problem 2**

LCP 3. 机器人大冒险

难度 中等 5 收藏 分享 切换为英文

题目描述 评论 (6) 题解(8) New 提交记录

力扣团队买了一个可编程机器人，机器人初始位置在原点 `(0, 0)`。小伙伴事先给机器人输入一串指令 `command`，机器人就会**无限循环**这条指令的步骤进行移动。指令有两种：

1. `U`：向 `y` 轴正方向移动一格
2. `R`：向 `x` 轴正方向移动一格。

不幸的是，在 xy 平面上还有一些障碍物，他们的坐标用 `obstacles` 表示。机器人一旦碰到障碍物就会被损毁。

给定终点坐标 `(x, y)`，返回机器人能否**完好**地到达终点。如果能，返回 `true`；否则返回 `false`。

示例 1：

```
输入：command = "URR", obstacles = [], x = 3, y = 2
输出：true
解释：U(0, 1) -> R(1, 1) -> R(2, 1) -> U(2, 2) -> R(3, 2)。
```

示例 2：

```
输入：command = "URR", obstacles = [[2, 2]], x = 3, y = 2
输出：false
解释：机器人在到达终点前会碰到(2, 2)的障碍物。
```

示例 3：

```
输入：command = "URR", obstacles = [[4, 2]], x = 3, y = 2
输出：true
解释：到达终点后，再碰到障碍物也不影响返回结果。
```

限制：

1. `2 <= command的长度 <= 1000`
2. `command` 由 `U`, `R` 构成，且至少有一个 `U`，至少有一个 `R`
3. `0 <= x <= 1e9, 0 <= y <= 1e9`
4. `0 <= obstacles的长度 <= 1000`
5. `obstacles[i]` 不为原点或者终点