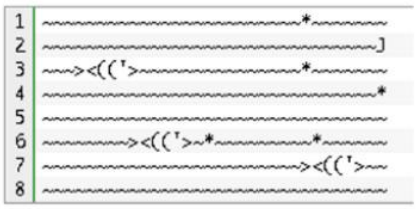
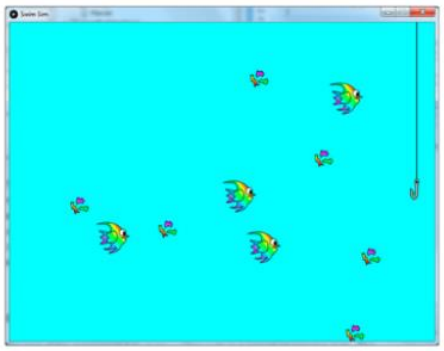
# 上机练习

这个项目需要你开发一个**鱼塘仿真软件**。鱼塘中的小鱼(用 ><(('> 表示)能够从左向右游（五条）。鱼饵(用 * 表示)一边下沉一边向左移动（五个）。另外会有一个鱼钩(用 J 表示)从鱼塘的底部被向上拉。

后续编程作业将在本项目的代码基础上继续下去，因此请保证代码干净、清晰、注释完整。以下是这个仿真软件的演示界面。





鱼塘仿真软件

以下为各变量含义：

| | | | |
|---|---|---|---|
| n | 用户输入鱼塘鱼的数 | m | 用户输入鱼饵的个数 |
| num | 记录时间 | Win | 记录鱼吃鱼饵的个数 |
| Win2 | 记录钓到鱼的个数 | Time1 | 用户输入规定时间 |
| Flagg | 记游戏状态，达成结局则跳出游戏 | Str[][] | 鱼塘 |
| HookplaceX<br>hookplaceY | 鱼钩坐标 | BaitplaceX<br>baitplaceY | 鱼饵坐标 |
| FishplaceX<br>fishplaceY | 鱼坐标 | Random_number | 鱼的位置基<br>配合 repetition |
| Repetition[]，s | 确保生成鱼在不同行，避免重叠 | w | 键盘输入鱼钩移动 |

思路：

整个软件的代码可以封装为几大函数

1. 主函数，包括初始化实参，随机生成鱼和鱼饵鱼钩坐标（并且保证不会出现覆盖等错误现象）各个物体的坐标和循环调用其余函数，配合 sleep 函数实现动态。

```
    int s = 0;
    random_number = rand() % 8;
    for (int j = 0; j < k; j++)
    {
        if (repetition[j] == random_number + 1)
            s = 1;
    }
    if (s == 0)
        break;
}
repetition[k] = random_number + 1;
fishplaceX[k] = random_number;
```

确保鱼在不同行，不会重叠

```
for (int k = 0; k < m; k++)
{ // 随机生成鱼饵的坐标
    baitplaceX[k] = 0;
    baitplaceY[k] = rand() % 32;
    while (str[baitplaceX[k]][baitplaceY[k]] != '~')
    {
        baitplaceY[k] += 3;
        baitplaceY[k] %= 32;
    }
}

bait(m, str, fishplaceX, fishplaceY, baitplaceX, baitplaceY, &win);

hookplaceY = rand() % 32;
while (str[hookplaceX - 1][hookplaceY] != '~')
{
    hookplaceY += 3;
    hookplaceY %= 32;
}
```

确保鱼饵和鱼钩不会初始在鱼身上，考虑特殊情况

2. Pool 鱼塘函数（~~），生成一个数字可在 define 中直接修改的鱼塘 str[][],另外为了实现鱼的尾巴在右边彻底游出池塘时，鱼头在从左边进入鱼塘，扩大数组的列 COL_，存放游出的鱼且保证数组不越界。

```
#define ROW 8   // 8行
#define COL_ 38 // 扩大鱼塘，实现鱼尾巴出去，头再出来的游动效果
#define COL 32  // 32列
```

3. Fish 鱼函数（><(('>），fish 函数调用其横坐标 fishplaceX ,纵坐标 fishplaceY.

实现鱼的游动：　鱼原本的位置变成海浪~

鱼的纵坐标++

现位置变为鱼游动后的字符

特殊情况特殊分析：　鱼的坐标是-1 就不打印（对应后来的鱼钩钓上鱼）

鱼的尾巴游出边界则在从左边又出

4. bait 鱼饵函数（*），调用鱼饵的横坐标 baitplaceX ,纵坐标 baitplaceY,鱼饵的运动方式是向左下方掉落，碰到鱼则被吃，掉落至底部则再从上方掉落

实现鱼饵掉落：　鱼饵原本位置变成海浪~

鱼饵的纵坐标--。横坐标++

现位置变为鱼饵*

特殊情况：　掉落至行底部，回到顶部，再次掉落，直至被吃

掉落至最左列，回到最右列，继续掉落，直至被吃

如果鱼饵坐标为-1.则跳过不再让此鱼饵运动（对应鱼饵被鱼吃）

实现鱼吃鱼饵：　鱼饵在掉落过程中倘若碰到鱼的身体任意部位或掉落至鱼嘴前，则鱼饵坐标置为-1（意味着鱼饵被吃消失），鱼饵被吃次数 win++

```
for (int j = 0; j < 6; j++)
{
    if (str[baitx[i]][baity[i]] == fishbody[j])
    {
        (*win)++;
        baitx[i] = -1;
        baity[i] = -1;
    }
}
if (str[baitx[i]][baity[i] - 1] == '>')
{
    (*win)++;
    baitx[i] = -1;
    baity[i] = -1;
    str[baitx[i]][baity[i]] = '>';
}
```

实现被吃就消失且不会影响鱼的运动

5. hook 鱼钩函数（J），同上调用鱼钩的横纵坐标 hookplaceX ,hookplaceY.鱼钩的运动方式是非固定键盘操作左右移动，上下移动固定方式从上往下循环移动

实现鱼钩运动：　倘若键盘敲 A，则左移，D，则右移，实现游戏互动

鱼钩在上下移动中循环，碰到底部则向上运动，碰到底部再向下运动

```
// 上下循环
if (flag_ == 0)
{
    (*hookplaceX)--;
    if ((*hookplaceX) == 0)
    {
        flag_ = 1;
    }
}
else if (flag_ == 1)
{
    (*hookplaceX)++;
    if ((*hookplaceX) == 7)
    {
        flag_ = 0;
    }
}
```

上下钓

实现鱼钩钓鱼：　鱼钩在运动中碰到除了~浪和鱼饵*之外的字符（即鱼的各个部位）则实现动态钓鱼，使鱼消失

具体过程：

```
if (str[(*hookplaceX)][(*hookplaceY)] != '~' && str[(*hookplaceX)][(*hookplaceY)] != '*')
{
    for (int i = 0; i < n; i++)
    {
        if ((*hookplaceX) == fishplaceX[i])
        {
            fishplaceX[i] = -1;
            break;
        }
    }
    (*win2)++;
    int flag = 0;
    int forward = (*hookplaceY) + 1;
    int backward = (*hookplaceY) - 1;
    if (str[(*hookplaceX)][(*hookplaceY)] == '>') // 鱼嘴或鱼尾
    {
        flag++;
    }

while (flag < 2)
{

    if (str[(*hookplaceX)][forward] != '~' && str[(*hookplaceX)][forward] != '*')
    {
        str[(*hookplaceX)][forward] = '~';
        forward++;
        if (str[(*hookplaceX)][forward] == '~' || str[(*hookplaceX)][forward] == '*')
            flag++;
    }
    if (str[(*hookplaceX)][backward] != '~' && str[(*hookplaceX)][backward] != '*' && backward != -1)
    {
        str[(*hookplaceX)][backward] = '~';
        backward--;
        if (backward == -1)
            flag++;
        if (backward != -1 && (str[(*hookplaceX)][backward] == '~' || str[(*hookplaceX)][backward] == '*'))
            flag++;
    }
}
```

1 动态找鱼的身体，flag 计数，forward 为向前找鱼嘴，backward 为向后找鱼尾

2 找到最左边鱼尾 flag+1，找到最右边鱼嘴 flag+1，鱼嘴鱼尾都是>字符，则可都记为‘>’，

3 直至 flag==2，即左右都找到，鱼的身体找全，让鱼的整体消失。

6. print 打印函数，实时更新鱼塘，实现动态效果，且在右下角显示鱼吃鱼饵次数，钓鱼次数和计时。

最后游戏的结尾有两种可能

1. 在自己输入的时间内钓完所有鱼，胜利！记录用时多少
2. 规定的时间结束，未钓完所有鱼，显示放生的鱼的条数


可优化部分：1.鱼吃鱼饵可将简单的计数换成其他效果，譬如生成小鱼，鱼身体变长（但可能需要另外的小鱼的数组，且小鱼长度不同，其他触发效果也需要添加相关触碰可能）