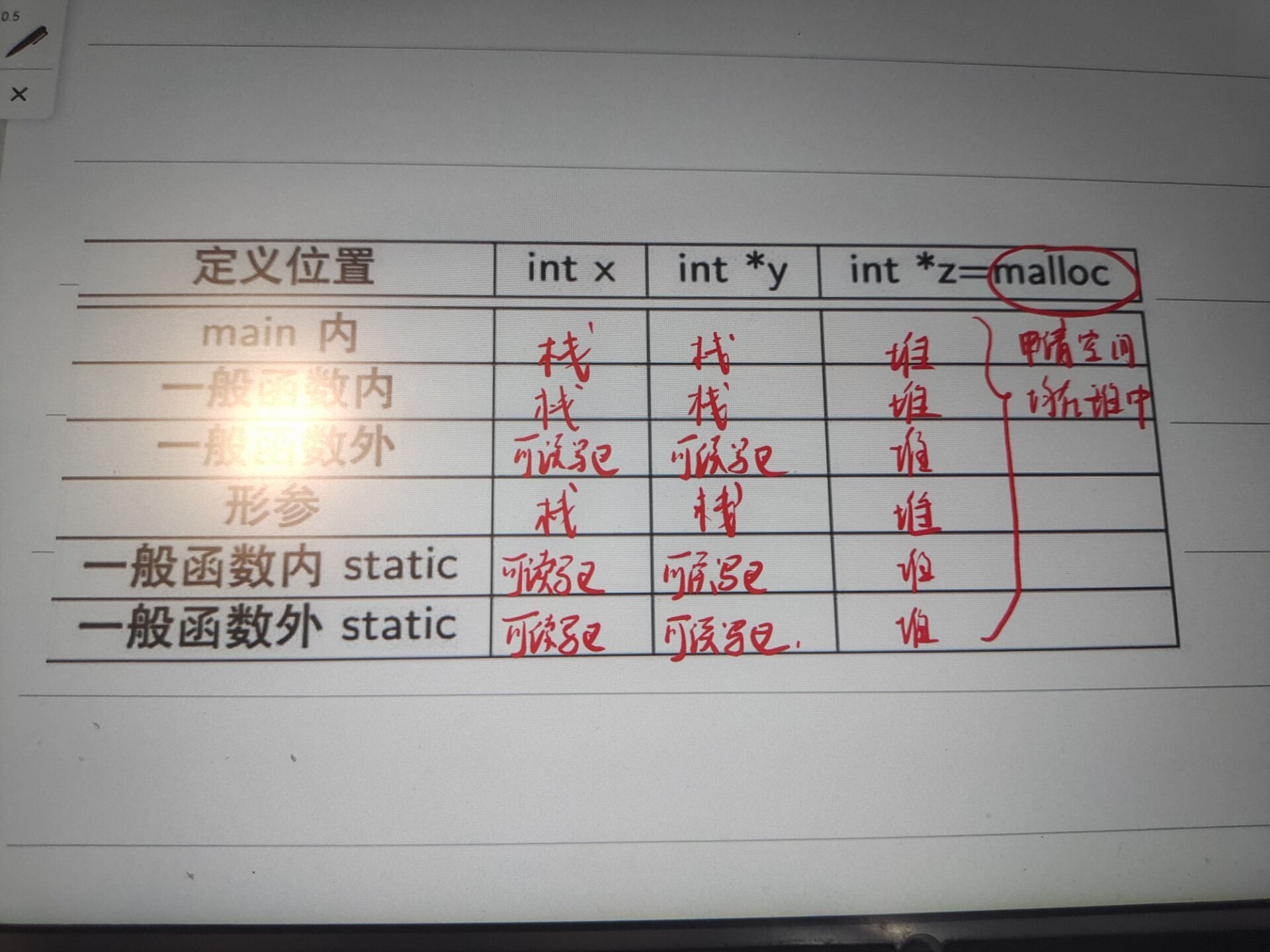
Hw1

填写表格：



最大子序列和问题：

代码如下：

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <ctype.h>

#define N\_MAX 1000

int main()

{

    int len;//数组长度

    int count=0;

    scanf("%d", &len);

    int num[N\_MAX];

    memset(num, 0, sizeof(num));

    for (int i = 0; i < len; i++)

    {

        scanf("%d", &num[i]);

    }

    int flag = 0;//判断数组是否存在正数，分情况处理

    int sum = 0;//记录当前子序列和

    int summax;//记录当前所计算的最大子序列和

    //判断是否存在正数

    for (int i = 0; i < len; i++)

    {

        if (num[i] > 0)

        {

            flag = 1;

            break;

        }

    }

    //有正数存在情况

    if (flag == 1)

    {

        summax=num[0];

        for (int i = 0; i < len; i++)

        {

            sum += num[i];

            count++;       /\*为了保证最大子序列和的子序列中序列长度至少为2，

                            因此每加一次计数，

                           在sum置为0时随之置为0，

                           如果为一则该最大值无效\*/

            if (sum > summax&&count!=1)

            {

                summax = sum;//每出现符合条件的更大子序列和，将其存入

            }

            if (sum < 0)/\*当前和小于0则该子序列必定不在最大子序列和中，

                         因此和为负数时重新开始求和\*/

            {

                sum = 0;

                count=0;

            }

        }

    }

    //无正数情况

    else//无正数情况最大序列和一定由两个相邻数求和，遍历求和存入最大数即可

    {

        summax=num[0]+num[1];

        for(int i=0;i<len;i++)

        {

            sum=num[i]+num[i+1];

            if(sum>summax)

            {

                summax=sum;

            }

            sum=0;

        }

    }

    printf("%d\n", summax);

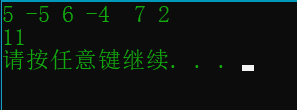
    system("pause");

    return 0;

}

运行结果举例：

第一个数表示待输入的序列的长度，后面的数表示序列，第二行是输出的最大子序列和



时间复杂度分析：

该算法的时间消耗主要分为两部分，一是没有正数的情况，二是存在正数的情况，没有正数时，只需遍历数组，依次求出两两相邻的数字的和，存入最大值即可，这样遍历一遍数组的时间复杂度为O（n）；存在正数时,将数组第一个数设为最大值，然后遍历数组，将最大和存入，如果和为负数，则从当前数字开始求和，这样只需遍历数组一遍便可求出最大子序列和，所以时间复杂度是O（n）；综上，整个算法的时间复杂度是O(n)

改进：如需求出对应的j，k的值，需要记录每次开始求和的位置，在存入summax中时，同时存入开始位置k以及当前位置j，在更新summax时同时更新k和j的值，最后一起输出即可。