内容回顾

第三章 结构化程序设计

- 基本语句：表达式语句; break; continue; return; ~~goto~~;
  - 尽量避免使用goto语句，滥用goto容易使程序逻辑混乱，可调试性变弱，程序运行效率变低
  - goto语句一般只在两种情况下使用：

    1. 跳出多层循环

```
for(;;)
{
    for(;;)
    {
        for(;;)
        {
            if (...)
            {
                goto _NEXT
            }
        }
    }
}
_NEXT:
...
```

    2. 异常处理

```c
int my_fun(char *str)
{
    int ret = -1;
    if(str == NULL)
    {
        goto _RET
    }
    ...
_RET:
    return ret;
}
```

- 程序结构

    1. 顺序结构
    2. 选择结构：if..else...; switch...case...;
    3. 循环结构：while...; do...while...; for...;

第四章 数组、字符串

- 数组在内存中为连续存储，二维数组按行存储
- 编程时注意边界情况，不要超出数组的索引范围
- 冒泡排序：原始；外循环早停；内循环缩小范围
- 多维数组：数组的数组
- 字符串：以'\0'为结尾的字符数组
- 常用字符串函数<string.h>

```c
unsigned int strlen(char* s);
char* strcpy(char* dest, const char* src);
char* strncpy(char* destinin, char* source, int maxlen);
char* strcat(char* dest, const char* src);
int strcmp(const char* str1, const char* str2);
//若str1=str2，则返回零；若str1<str2，则返回负数；若str1>str2，则返回正数
char* strlwr(char* s);
char* strupr(char* s);
//兼容性说明：strlwr和strupr不是标准C库函数，只能在VC中使用。linux gcc环境下需要自行定义这个函数。
int puts(const char* string);
char* gets(char* string);
//读入成功，返回与参数buffer相同的指针；读入过程中遇到EOF(End-of-File)或发生错误，返回NULL指针。因此遇到返回值为NULL的情况，要用ferror或feof函数检查是发生错误还是遇到EOF。
//gets()可以无限读取，易发生溢出。如果溢出，多出来的字符将被写入到堆栈中，破坏了堆栈原先的内容
```

第五章 函数

- 函数定义
- 函数调用
- 值传递：形参和实参在数值上一致，但各自的内存地址不同，其指针不同。实际上形参为函数内部新建的局部变量
- 函数嵌套使用：先调用的后返回值，栈结构
- 函数说明：调用后定义的函数，调用其他文件定义的函数
- 递归函数：函数直接或间接地调用自身

- 递归函数模板
    1. 首先确定边界情形，即定义递归函数出口
    2. 构建递归逻辑，即如何将较大的解空间缩小为较小的解空间，缩小求解规模
    3. 递归栈空间存在上限，必要时结合栈的数据结构重写为迭代形式
- 存储类别：
    - auto：局部变量，默认自动存储期。离开定义范围值消失。
    - register：局部变量，将变量装载到寄存器中，加快运行速度。寄存器数量有限，编译器可能忽略。
    - extern：全局变量和函数，本文件和其他文件可见。
    - static：全局变量，仅本文件可见。局部变量，定义的函数范围内可见，且函数运行结束时值保留
- 编译预处理命令
    1. 宏定义

```
#define True 1

#define SQR(x) ((x)*(x))

#undef TRUE
#undef SQR(x)
```

    3. 文件包含

```
#include <stdio.h>
#include "my_prog.h"
```

    5. 条件编译
        1. #ifdef... #else... #endif
        2. #ifndef... #else... #endif
        3. #if... #else... #endif

习题讲解

c语言标准

- 本课程包括最终的考试仅使用C89标准
- 请勿使用C99标准或C11标准，具体为：
    1. 复合语句内所有变量的定义应在printf()，scanf()等语句之前
    2. 定义数组时，数组长度不能为变量，即不要使用动态数组

```
int n;
scanf("%d", &n);
int arr[n] ;
```

程序调试主要步骤

1. 编译失败，通过下方的错误列表窗口按顺序依次排除错误
2. 编译成功，运行失败，检查程序中是否有超出数组索引的情形
3. 运行成功，结果错误，在适当的地方插入断点，通过下方的局部变量窗口或手动添加监视，查看中间变量或表达式的值是否符合预期，以此判断程序逻辑有无出错。根据实际需要选择“逐语句”“逐过

程”“跳出”“继续”。
4. 结果与题目中的例子相符，考虑边界情形，如极大极小值，有序无序数组，错误输入能否抛出异常等，提升程序的鲁棒性

39-7

```
i=2 c=Bk=123456f=5.8 x=3.4
```

40-10

- 语句不够简洁

5小题

- 整型运算和实型运算之间的切换

```
y = (int)(x * 1000 + 0.5) / 1000.
```

40-11

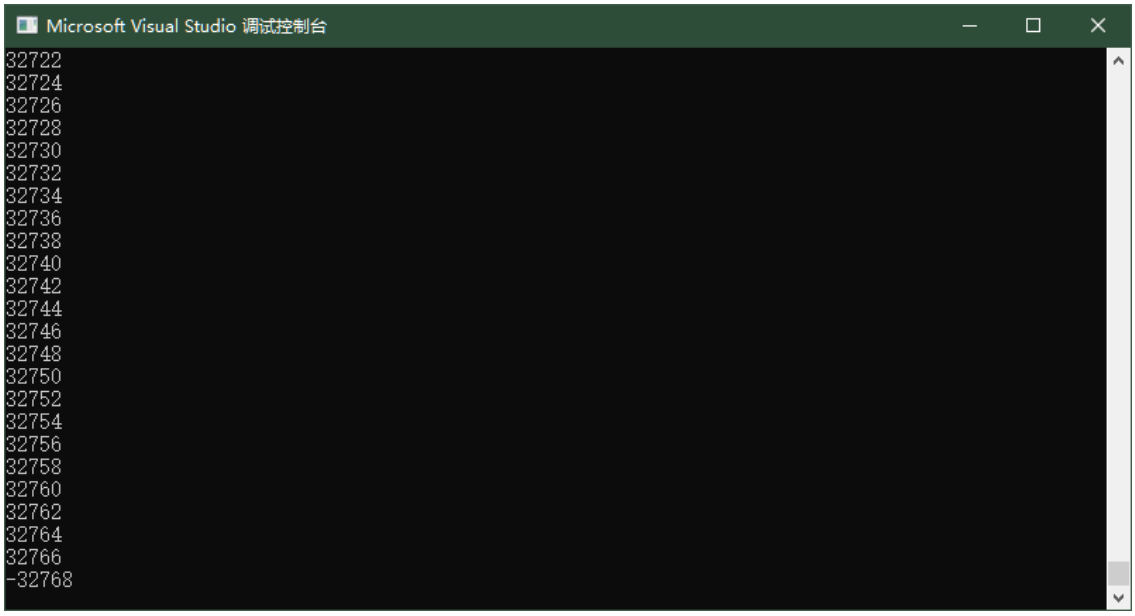
- `^`在C中为按位异或，在MATLAB中为幂运算
- <math.h> `double pow(double x, double y)`

40-14

- 整数除法应当考虑零除错误的情形

64-8

- 对于short int，输出32766后变为-32768；
- 对于int，输出2147483646后变为-2147483648



64-11

- 最后几行出现没有对齐的情况

```
for (space = 40; ; space -= 2)
{
    printf("%*c", space, ' ');
    ...
}
```

- 当space=0时，最小位宽为0而`' '`本身占据了一个位宽，因此位宽限制不起作用.。直观上看，该行向右移动一格

65-15

```
float x;
long int i, i_max = 1e6;
long double res = 0., tmp;
printf("please input a floating number: ");
scanf("%f", &x);
for (i = 0; i < i_max; i++)
{
    if (i == 0)
    {
        tmp = 1;
        res += tmp;
        continue;
    }
    tmp = tmp * x / i;
    res += tmp
}
printf("result: %.10f\n", res);
printf("gt:     %.10f", exp(x));
return 0;
```

- 直接计算x^n和$n!$容易溢出，超出浮点数表示范围，阶乘计算时应警惕这一点；
- 随着n增大每一项的值$x^n/n!$应随之减小，理论上每一项不会溢出
- 因此可以在前一项的基础上进行更新，即$x^n/n! = x^{n-1}/(n-1)! * x/n$

64-20

- 输入要求为整数，而非整数数组或字符串

89-10

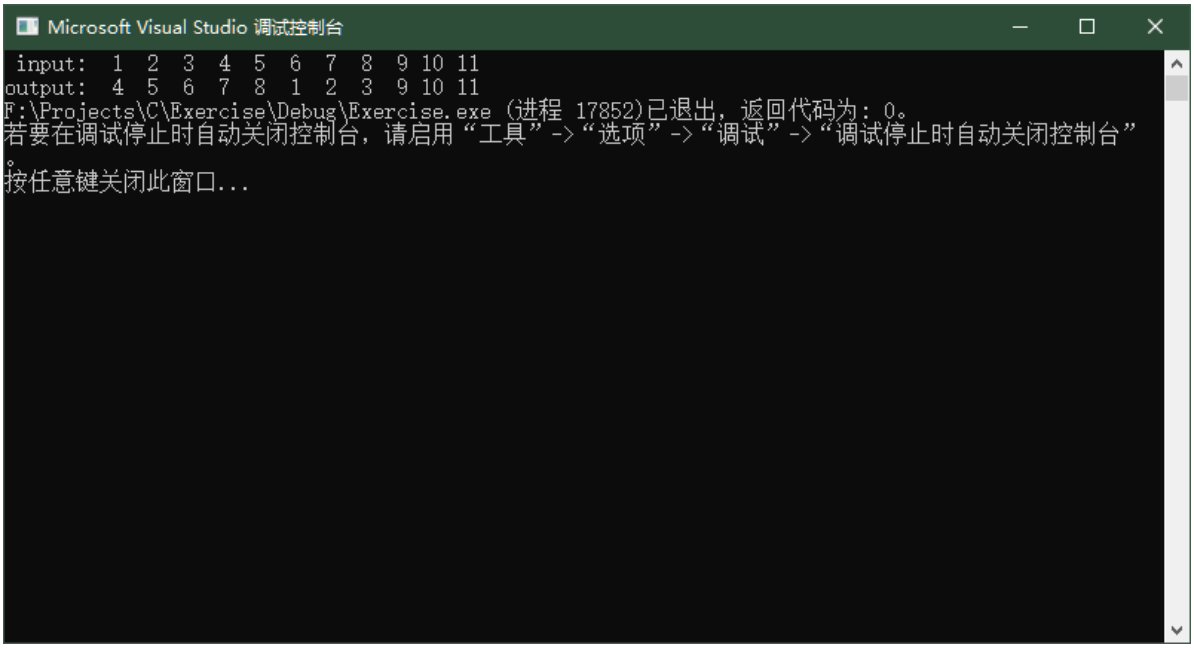
- 主要思路：
    1. 首先考虑特殊情况，即当前排列为最大，显然该排列应为降序排列
    2. 考虑一般情形，从右往左看，当其不符合升序排列时，说明这个子集存在更大的表示
    3. 将子集划分为待交换数字和降序排列，在降序排列中找到大于待交换数字的最小值并将两者交换
    4. 将交换后的数组原来降序排列位置上的数字按升序排列

89-12

- 未考虑数组长度>n的情况
- 交换数组元素时破坏了后续数组元素

```
#include<stdio.h>
```

```c
# define N 11

void reverse(int* arr, int left, int right)
{
    int i;
    int tmp;
    for (i = 0; i <= (right - left) / 2; i++)
    {
        tmp = arr[left + i];
        arr[left + i] = arr[right - i];
        arr[right - i] = tmp;
    }
}

int main()
{
    int arr[N] = { 1,2,3,4,5,6,7,8,9,10,11 };
    int i;
    int m = 3, n = 8;

    printf(" input: ");
    for (i = 0; i < N; i++)
    {
        printf("%2d ", arr[i]);
    }

    reverse(arr, 0, n - 1);
    reverse(arr, 0, n - 1 - m);
    reverse(arr, n - m, n - 1);

    printf("\noutput: ");
    for (i = 0; i < N; i++)
    {
        printf("%2d ", arr[i]);
    }

    return 0;
}
```



89-14

- 主要是观察方阵的规律

  1. 反对角线的行坐标和列坐标的和相同，和为奇数升序，和为偶数降序

```
1   3   4   10
2   5   9   11
6   8   12  15
7   13  14  16
```

  2. 方阵可一层层拆解，划分为更小的相同规律的方阵，另外需要考虑N奇偶是否有差异

```
1   2   3   4
12  13  14  5
11  16  15  6
10  9   8   7
```

89-16

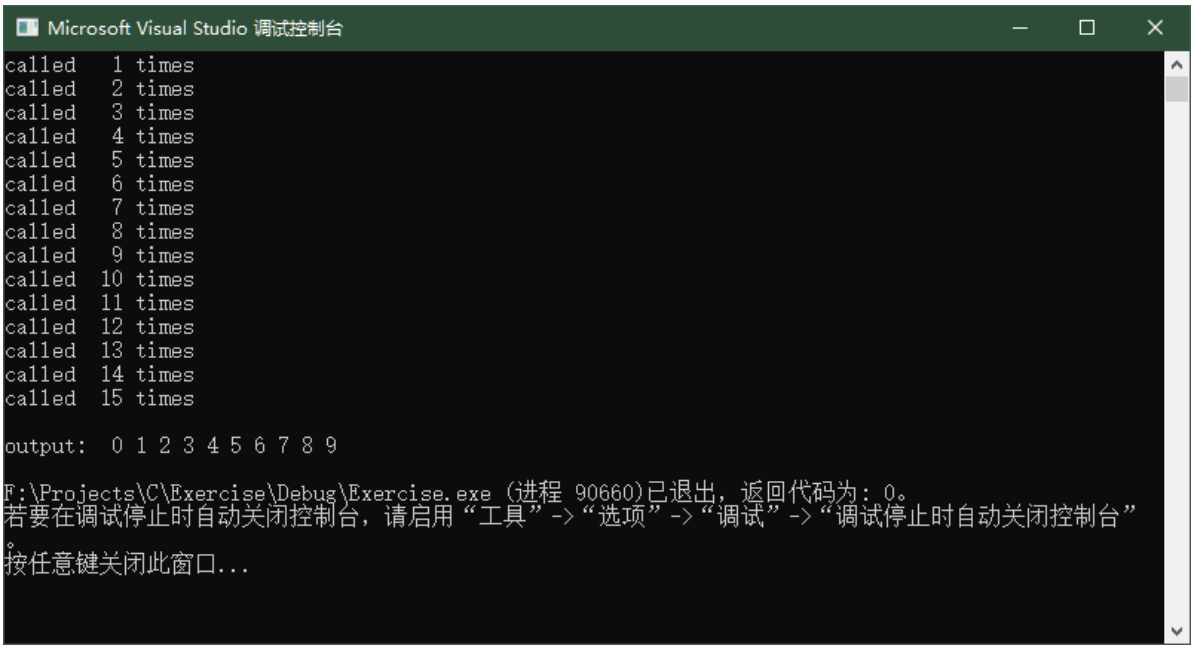
- 主要思路

  1. 遍历字符串，忽略前导空白符，读取第一个非空白符
  2. 记录当前整理后字符串的有效长度，为了删除后随空白符
  3. 当前字符为非空白符，有效长度加一；当前字符为空白字符且上一字符非空白，有效长度不变，直至遍历到下一非空白符，有效长度加二，即空白符+非空白符。
  4. 字符数组最后补充'\0'，构成字符串

11/12-1

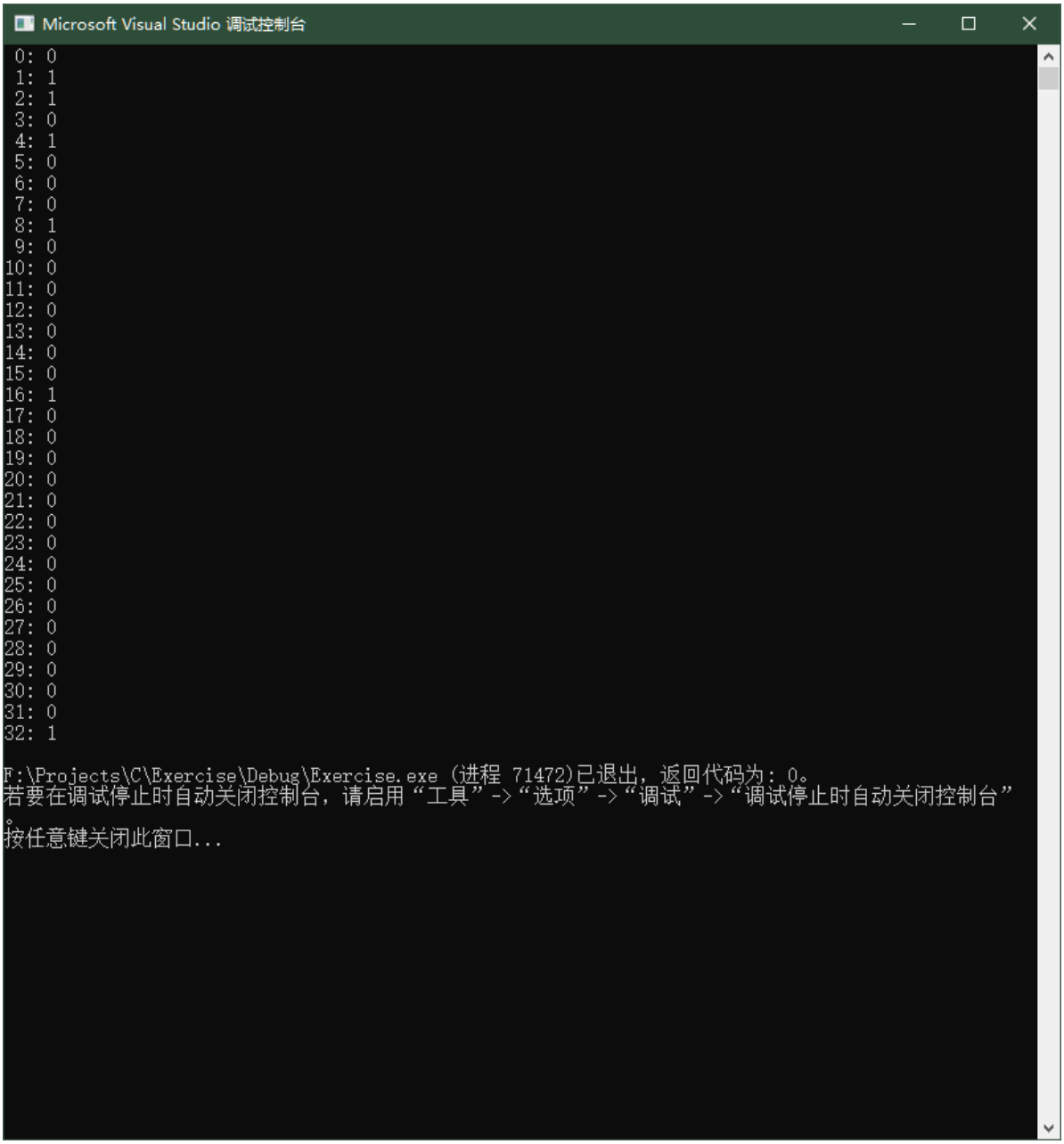
```c
#include<stdio.h>

//地址传递 或 使用全局变量的数组
void exchange(int* a, int j)
{
    static cnt_fun = 0;
    int temp;
    printf("called %3d times\n", ++cnt_fun);
    temp = a[j];
    a[j] = a[j - 1];
    a[j - 1] = temp;
    return;
}

int main()
{
    int a[10] = { 1,2,7,3,5,6,4,8,9,0 };
    int n = 10;
    int i, j;
    for (i = 0; i < 10; i++)
    {
        for (j = n - 1; j > i; j--)
        {
            if (a[j] < a[j - 1])
            {
                exchange(a, j);
```

```
            }
        }
    }
    printf("\noutput: ");
    for (i = 0; i < n; i++)
    {
        printf("%2d", a[i]);
    }
    printf("\n");
    return 0;
}
```



11/12-2

```
#include<stdio.h>
#define JUDGE(x) ((x) <= 0 ? 0 : (((x) & (x-1)) == 0))
#define N 2<<4

int main()
{
    int n;

    for (n = 0; n <= N; n++)
    {
        printf("%2d: %d\n", n, JUDGE(n));
    }
    return 0;
}
```

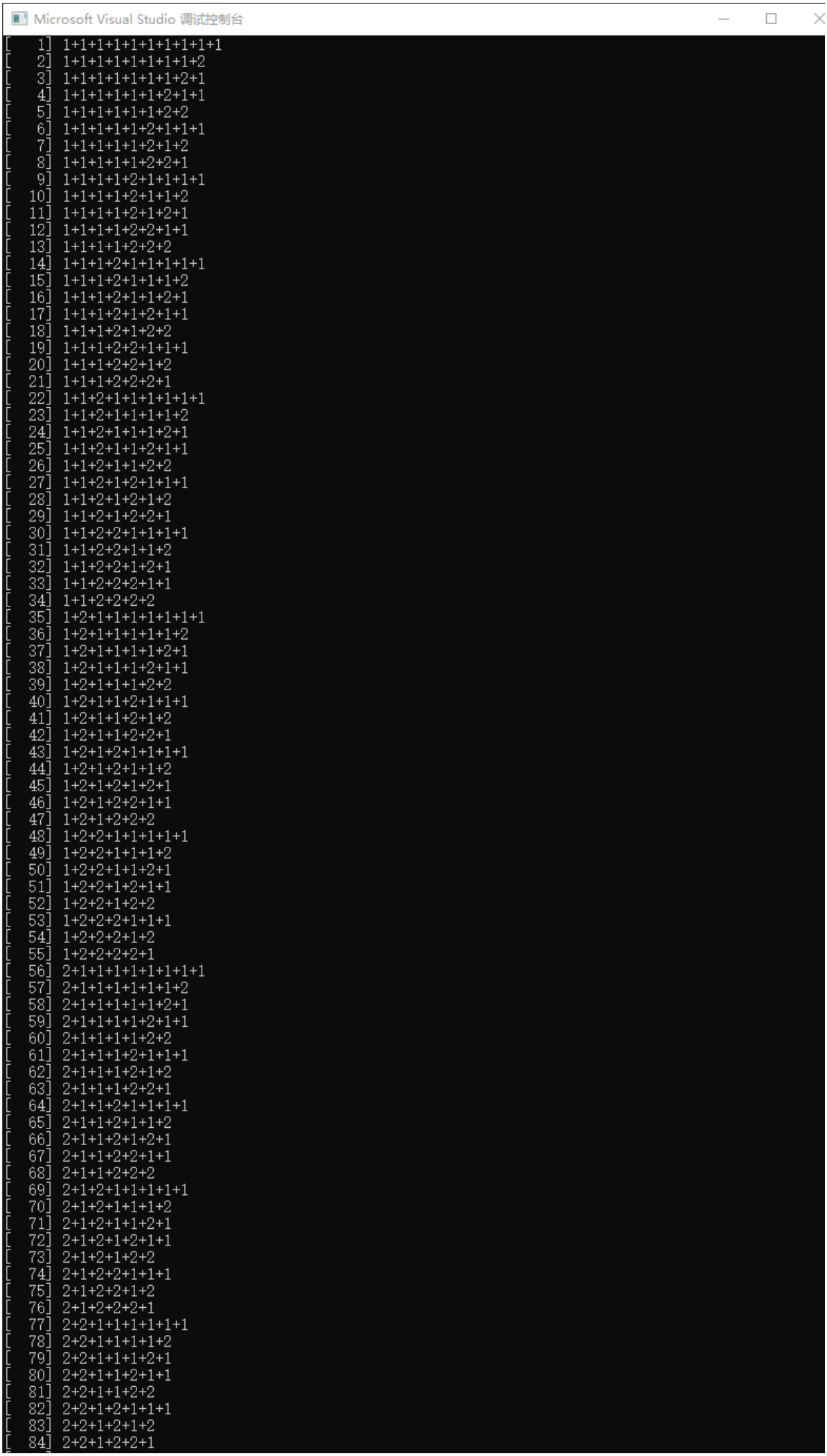



```c
#include<stdio.h>
#define JUDGE(x, res)   {\
                            while(1){\
                                if(x%2 == 1 && x/2 == 0){\
                                    res=1;\
                                    break;\
                                }\
                                if(x%2 == 0 && x/2!=0){\
                                    x /= 2;\
                                }else{\
                                    res=0;\
                                    break;\
                                }\
                            }\
                        }
#define N 2<<4

int main()
{
    int n;
    int res;
```

```
    int x;

    for (n = -N; n <= N; n++)
    {
        x = n;
        JUDGE(x, res)
        printf("%4d: %d\n", n, res);
    }
    return 0;
}
```

11/12-3

```
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
#include<string.h>
#define N 2<<10

int cnt = 0;

int helper(int n, char* explain)
{
    int res1, res2;
    char exp1[N], exp2[N];
    strcpy(exp1, explain);
    strcpy(exp2, explain);
    if (n == 1)
    {
        printf("[%4d] %s\n", ++cnt, strcat(exp1, "1"));
        return 1;
    }
    else if (n == 2)
    {
        printf("[%4d] %s\n", ++cnt, strcat(exp1, "1+1"));
        printf("[%4d] %s\n", ++cnt, strcat(exp2, "2"));
        return 2;
    }
    res1 = helper(n - 1, strcat(exp1, "1+"));
    res2 = helper(n - 2, strcat(exp2, "2+"));
    return res1 + res2;
}

int main()
{
    int n = 10;
    char explain[N] = "";
    int res = helper(n, explain);
    printf("\n%d\n", res);
}
```


Microsoft Visual Studio 调试控制台
[1] 1+1+1+1+1+1+1+1+1+1
[2] 1+1+1+1+1+1+1+1+2
[3] 1+1+1+1+1+1+1+2+1
[4] 1+1+1+1+1+1+2+1+1
[5] 1+1+1+1+1+1+2+2
[6] 1+1+1+1+1+2+1+1+1
[7] 1+1+1+1+1+2+1+2
[8] 1+1+1+1+1+2+2+1
[9] 1+1+1+1+2+1+1+1+1
[10] 1+1+1+1+2+1+1+2
[11] 1+1+1+1+2+1+2+1
[12] 1+1+1+1+2+2+1+1
[13] 1+1+1+1+2+2+2
[14] 1+1+1+2+1+1+1+1+1
[15] 1+1+1+2+1+1+1+2
[16] 1+1+1+2+1+1+2+1
[17] 1+1+1+2+1+2+1+1
[18] 1+1+1+2+1+2+2
[19] 1+1+1+2+2+1+1+1
[20] 1+1+1+2+2+1+2
[21] 1+1+1+2+2+2+1
[22] 1+1+2+1+1+1+1+1+1
[23] 1+1+2+1+1+1+1+2
[24] 1+1+2+1+1+1+2+1
[25] 1+1+2+1+1+2+1+1
[26] 1+1+2+1+1+2+2
[27] 1+1+2+1+2+1+1+1
[28] 1+1+2+1+2+1+2
[29] 1+1+2+1+2+2+1
[30] 1+1+2+2+1+1+1+1
[31] 1+1+2+2+1+1+2
[32] 1+1+2+2+1+2+1
[33] 1+1+2+2+2+1+1
[34] 1+1+2+2+2+2
[35] 1+2+1+1+1+1+1+1+1
[36] 1+2+1+1+1+1+1+2
[37] 1+2+1+1+1+1+2+1
[38] 1+2+1+1+1+2+1+1
[39] 1+2+1+1+1+2+2
[40] 1+2+1+1+2+1+1+1
[41] 1+2+1+1+2+1+2
[42] 1+2+1+1+2+2+1
[43] 1+2+1+2+1+1+1+1
[44] 1+2+1+2+1+1+2
[45] 1+2+1+2+1+2+1
[46] 1+2+1+2+2+1+1
[47] 1+2+1+2+2+2
[48] 1+2+2+1+1+1+1+1
[49] 1+2+2+1+1+1+2
[50] 1+2+2+1+1+2+1
[51] 1+2+2+1+2+1+1
[52] 1+2+2+1+2+2
[53] 1+2+2+2+1+1+1
[54] 1+2+2+2+1+2
[55] 1+2+2+2+2+1
[56] 2+1+1+1+1+1+1+1+1
[57] 2+1+1+1+1+1+1+2
[58] 2+1+1+1+1+1+2+1
[59] 2+1+1+1+1+2+1+1
[60] 2+1+1+1+1+2+2
[61] 2+1+1+1+2+1+1+1
[62] 2+1+1+1+2+1+2
[63] 2+1+1+1+2+2+1
[64] 2+1+1+2+1+1+1+1
[65] 2+1+1+2+1+1+2
[66] 2+1+1+2+1+2+1
[67] 2+1+1+2+2+1+1
[68] 2+1+1+2+2+2
[69] 2+1+2+1+1+1+1+1
[70] 2+1+2+1+1+1+2
[71] 2+1+2+1+1+2+1
[72] 2+1+2+1+2+1+1
[73] 2+1+2+1+2+2
[74] 2+1+2+2+1+1+1
[75] 2+1+2+2+1+2
[76] 2+1+2+2+2+1
[77] 2+2+1+1+1+1+1+1
[78] 2+2+1+1+1+1+2
[79] 2+2+1+1+1+2+1
[80] 2+2+1+1+2+1+1
[81] 2+2+1+1+2+2
[82] 2+2+1+2+1+1+1
[83] 2+2+1+2+1+2
[84] 2+2+1+2+2+1

```
[  85] 2+2+2+1+1+1+1
[  86] 2+2+2+1+1+2
[  87] 2+2+2+1+2+1
[  88] 2+2+2+2+1+1
[  89] 2+2+2+2+2

89

F:\Projects\C\Check\Debug\Check.exe (进程 81740)已退出，返回代码为：0。
```