考纲

考试内容和要求
（一）、考试要求：
1.掌握结构化程序设计方法或面向对象编程技术
2.掌握程序语言的基础知识
3.掌握常用算法设计及描述方法
4.掌握程序调试方法
5.具有阅读程序和改错能力
6.具有良好的编程风格
7.用 C、C++或其它语言熟练编写程序
（二）、考试内容：
1.数据类型及其操作：基本数据类型、数组、指针、结构体、链表等的定义、初始化、引用和操作
2.程序语言的三种控制结构：顺序、选择、循环
3.程序输入输出实现：程序中赋值、键盘输入和输出，通过文件进行数据存取
4.函数：函数定义、函数调用、参数传递、函数返回
5.算法描述方法：程序流程图、N-S 盒图、伪代码等
6.常用算法示例：
（1）加法器与累乘器
（2）求最大数与最小数
（3）排序（冒泡排序、选择排序等）
（4）大小写字母转换
（5）判别键盘输入字符的类别
（6）判别闰年
（7）百分制成绩与等级制成绩互相转换
（8）求两个数的最大公因数和最小公倍数
（9）求菲比拉契数列有限项
（10）统计学生成绩，包括总成绩、平均成绩、各分数段人数等
（11）验证哥德巴赫猜想
（12）用穷举法求某数段的素数、水仙花数、完全平方数等
（13）求近似数（如定积分、用牛顿迭代法或二分法或弦截法求多元方程的根）
（14）求两个矩阵之和、之积
（15）统计输入字符中的单词个数

1.输出螺旋矩阵，试题标准答案

思路是四个边分别输出

```
#include<stdio.h>
using namespace std;
int main()
{
    int i, j, k = 0, n, a[10][10], m;
    printf("Enter n(n<10)");
    scanf_s("%d", &n);
    if ((n % 2) == 0)
        m = n / 2;
    else
        m = n / 2 + 1;

    for (i = 0; i <m; i++)//按螺旋方阵的层数循环，从最外层一层层开始输出
```

```
    {
        for (j = i; j < n - i; j++)
            a[i][j] = ++k;
        for (j = i + 1; j < n - i; j++)
            a[j][n - i - 1] = ++k;
        for (j = n - i - 2; j >= i; j--)
            a[n - i - 1][j] = ++k;
        for (j = n - i - 2; j >= i + 1; j--)
            a[j][i] = ++k;
    }
    for (i = 0;i<n;i++)
    {
        for (j = 0;j<n;j++)
            printf("%5d", a[i][j]);
        printf("\n");
    }

}
```

自己胡写的另一种思路

方法是类似搜索

```
#include<iostream>
using namespace std;
int main()
{
    int dir[4][2] = { 0,1,1,0,0,-1,-1,0 };
    int map[12][12];
    int n;
    while (scanf_s("%d", &n) != EOF)
    {
        int ni = 0, nj = 0;
        int d = 0;
        memset(map, 0, sizeof(map));
        map[0][0] = 1;
        int stepLeft = n*n - 1, step = 1;
        while (stepLeft)
        {
            int _ni = ni + dir[d][0], _nj = nj + dir[d][1];
            if (_ni<0 || _ni>=n || _nj<0 || _nj>=n || map[_ni][_nj] != 0)
            {
                d = (d + 1) % 4;
            }
            ni += dir[d][0];
            nj += dir[d][1];
            map[ni][nj] = ++step;
            stepLeft--;
        }
        for (int i = 0;i < n;i++)
        {
            for (int j = 0;j < n;j++)
            {
                printf("%5d", map[i][j]);
            }
            cout << endl;
        }
    }
```

```
    return 0;
}
```

2.牛顿迭代法求根

2x^3-4x^2+3*x-6=0 在1.5附近的根

思路，随便画一个曲线，设1.5为x0，得到f(x0)，过x,f(x0)做曲线的切线，再得到切线过x轴的交点x1，检查x0与x1的距离，如果不够小，那么将x1作为新的x0重复上述的计算。具体怎么算，画个图求个导就出来了。

```
#include<iostream>
#include<stdio.h>
#include<math.h>
using namespace std;
int main()
{
    double x0 = 1.5, x;
    double f0, k,;
    while (1)
    {
        f0 = 2 * x0*x0*x0 - 4 * x0*x0 + 3 * x0 - 6;
        k = 6 * x0*x0 - 8 * x0 + 3;
        x = x0 - f0 / k;
        if (fabs(x0 - x) < 1e-5)break;
        else x0 = x;
    }
    cout << x;
    return 0;
}
```

3.c语言读写操作相关（以前搬砖基本不用，做个备忘）

代码功能是统计txt文本中的字符个数

```
#include<iostream>
#include<stdlib.h>
using namespace std;
int main()
{
    FILE *fp;
    int num = 0;
    char c;
    if ((fp = fopen("D:\\in.txt", "r")) == NULL)
    {
        printf("Open error\n");
        exit(0);
    }
    c = fgetc(fp);
    while (c != EOF)
    {
        num++;
        c = fgetc(fp);
    }
    cout << num << endl;
    fclose(fp);
    return 0;
```

```
}
```

4.欧几里得辗转相除法，算最小公倍数和最大公因数，待分析

```
#include<iostream>
#include<stdlib.h>
using namespace std;
int main()
{
    int x, y, r, a, b;
    scanf("%d%d", &x, &y);
    a = x;
    b = y;
    if (x>y)
        r = x, x = y, y = r;
    r = x%y;
    while (r)
    {
        x = y;
        y = r;
        r = x%y;
    }
    printf("%d %d", y, a*b / y);
    return 0;
}
```

5.汉诺塔，递归解法

思路，长话短说，把一次操作抽象出来，分为abc三个柱子，视A为满，bc为空，a上的分为上下两部分。显然这次的操作是把a的上部分绕过c放到b上，再把a的下部分放到c上，最后把b上的绕过a放到c。不清楚的话自己摆几个东西试试。

```
#include<stdio.h>
#include<iostream>
#include<math.h>
using namespace std;
int cnt = 0;
void Move(char a, char c)
{
    cnt++;
}
void Hannoi(int n, char a, char b, char c)
{
    if (n == 1)Move(a, c);
    else
    {
        Hannoi(n - 1, a, c, b);
        Move(a, c);
        Hannoi(n - 1, b, a, c);
    }
}
int main()
{
    Hannoi(3, 'a', 'b', 'c');
    cout << cnt << endl;
    return 0;
```

```
}
```

6.二分法求方程解

方程为2*x*^3-4*x*^2+3*x-6=0，范围[-10,10]

思路，先得到左值坐标和右值坐标，计算两点中间的坐标，如果中间值跟右值同号，说明零点在左边，那就把中间值作为新的右值，反之则反之，依次循环。只要中间值y坐标接近与0就跳出循环了。

```
#include<stdio.h>
#include<iostream>
#include<math.h>
using namespace std;
float f(float x)
{
    return 2 * x*x*x - 4 * x*x + 3 * x - 6;
}
int main()
{
    float ym, yl, yr, middle, left, right;
    cin >> left >> right;
    yr = f(right);
    yl = f(left);
    do
    {
        middle = (left + right) / 2;
        ym = f(middle);
        if (yr*ym > 0)
        {
            right = middle;
            yr = ym;
        }
        else
        {
            left = middle;
            yl = ym;
        }
    } while (fabs(ym) >= 1e-6);
    cout << middle << endl;
    return 0;
}
```

7.几种简单的排序

①选择排序，傻瓜式排序，假设n个元素存在一个数组里，第一次找后n个里最大的跟下标0元素交换，第二次找后n-1个里最大的跟下标1元素交换，依次类推，代码略

②冒泡，就记住两行即可，原理没啥好说的，大意就是两个相邻的比较交换，背过就完事了

```
int a[5] = { 6,2,4,1,7 };
int len=sizeof(a)/sizeof(int);
for (int i = 0;i < len -1;i++)
{
    for (int j = 0;j < len-i-1;j++)
    {
        if (a[j] > a[j + 1])
        {
            swap(a[j], a[j + 1]);
        }
    }
}
```

③双向冒泡，原理就是正着交换一遍，倒着交换一遍，每换一遍就修改终点位置，flag用于判断是不是所有的元素都按照顺序排列完毕

```
    int a[5] = { 6,2,4,1,7 };
    int len=sizeof(a)/sizeof(int);
    int right= len -1, left=0;
    bool flag = true;

    while (flag)
    {
        flag = false;
        for (int i = left;i < right;i++)
        {
            if (a[i] > a[i + 1])
            {
                swap(a[i], a[i + 1]);
                flag = true;
            }
        }
        --right;
        for (int j = right;j > left;j--)
        {
            if (a[j] < a[j - 1])
            {
                swap(a[j], a[j - 1]);
                flag = true;
            }
        }
        ++left;
    }

    for (int i = 0;i < len;i++)
        cout << a[i] << endl;
```

④插入排序，待分析

```
    int a[5] = { 6,2,4,1,7 };
    int len=sizeof(a)/sizeof(int);

    for (int i = 1;i < len;i++)
    {
        int temp = a[i], j = i;
        if (a[j - 1] > temp)
```

```
        {
            while (j >= 1 && a[j - 1] > temp)
            {
                a[j] = a[j - 1];
                j--;
            }
        }
        a[j] = temp;
    }

    for (int i = 0;i < len;i++)
        cout << a[i] << endl;
```

8.魔方阵，数学题，放弃

9.折半查找，跟二分法求函数根一个思路，不解释了

递归法

```
int a[10];
void Bsearch(int left, int right, int n)
{
    int middle = (left + right) / 2;
    if (left>right)
    {
        cout << "not found" << endl;
    }
    else if (a[middle] == n)
    {
        cout << middle << endl;
    }
    else if(n>a[middle])
    {
        Bsearch(middle + 1, right, n);
    }
    else if(n<a[middle])
    {
        Bsearch(left, middle - 1, n);
    }
}
```

循环法

```
#include<stdio.h>
#include<iostream>
using namespace std;
int a[10] = { 0,1,2,3,4,5,6,7,8,9 };
int Bsearch(int n)
{
    int left = 0, right = 9,middle;
    while (left <= right)
    {
        middle = (left + right) / 2;
        if (n==a[middle])
            return middle;
        else if(n>a[middle])
            left = middle + 1;
```

```
        else if (n < a[middle])
            right = middle - 1;
    }
    return -1;
}
int main()
{
    cout<<Bsearch(7)<<endl;
    return 0;
}
```

10.递归小练，int转字符串

```
char c[100];
void Func(int deepth, int n, int cnt)//递归深度，数字，数字位数
{
    if (deepth == cnt)
    {
        c[deepth] = '\0';
        return;
    }
    c[cnt - deepth - 1] = char(n % 10 + '0');
    Func(deepth + 1, n / 10, cnt);
}
```

11.给出年月日，计算这是该年的哪一天

```
int main()
{
    int year, day, month,days;
    while (cin >> year >> month >> day)
    {
        days = 0;
        int months[13] = { 0,31,28,31,30,31,30,31,31,30,31,30 };
        if (year % 400 == 0 || (year % 4 == 0 && year % 100 != 0))
        {
            months[2] = 29;
        }
        for (int i = 1;i < month;i++)
        {
            days += months[i];
        }
        cout << days+day << endl;
    }
    return 0;
}
```

12.n个人围成一圈，顺序拍号，从第一个人开始报数（从1到3报数），凡报到3的人退出圈子，问最后留下来的是原来第几号的那位

数组版

```
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
using namespace std;
```

```
int main()
{
    bool mark[10];//最大就10
    int n, cnt, index;
    int num;
    while (cin >> n)
    {
        index=cnt = 0;
        num = 1;
        memset(mark, false, sizeof(mark));
        while (cnt < n-1)
        {
            if (mark[index] == false&&num>=3)
            {
                mark[index]=true;
                cnt++;
                num = 1;
            }
            else if(mark[index]==false)
                num++;
            index=(index+1)%n;
        }
        for (int i = 0;i < n;i++)
        {
            if (mark[i]==false)
                cout << i+1 << endl;
        }
    }
    return 0;
}
```

循环双向链表版

```
#include<iostream>
#include<stdio.h>
using namespace std;
struct Node
{
    Node* next;
    Node* pre;
    int index;
};
int main()
{
    int n;
    while (cin >> n)
    {
        Node *h = (Node*)malloc(sizeof(Node));
        h->index = 1;
        Node* p=h;
        for (int i = 2;i <= n;i++)
        {
            Node* node = (Node*)malloc(sizeof(Node));
            node->index = i;
            p->next = node;
            node->pre = p;
            p = p->next;
```

```
        }
        p->next = h;
        h->pre = p;

        int cnt = 0,time=1;
        p = h;
        while (cnt < n-1)
        {
            if (time == 3)
            {
                Node* dp = p;
                p->pre->next = p->next;
                p->next->pre = p->pre;
                p = p->next;
                free(dp);
                time = 1;
                cnt++;
            }
            else
            {
                time++;
                p = p->next;
            }
        }
        cout << p->index <<endl;
    }
    return 0;
}
```

13.斐波那契数列

递归版

```
#include<iostream>
#include<stdio.h>
using namespace std;
int F(int n)
{
    if (n == 1 || n == 2)
        return 1;
    else
        return F(n - 1) + F(n - 2);
}
int main()
{
    int n;
    while(cin>>n)
        cout << F(n)<<endl;
    return 0;
}
```

循环版

```
#include<iostream>
#include<stdio.h>
using namespace std;
int main()
```

```
{
    int n;
    while (cin >> n)
    {
        int n1 = 1, n2 = 1, n3;
        if (n <= 2)
        {
            cout << 1 << endl;continue;
        }
        else
        {
            for (int i = 3;i <= n;i++)
            {
                n3 = n1 + n2;
                n1 = n2;
                n2 = n3;
            }
        }
        cout << n3 <<endl;
    }
    return 0;
}
```

14.穷举法

水仙花

```
bool Narcissistic(int n)
{
    int n1= n % 10, n2= n / 10 % 10, n3= n / 100 % 10;
    return n1*n1*n1+n2*n2*n2+n3*n3*n3== n;
}
```

素数

```
bool Prime(int n)
{
    for (int i = 2;i <= sqrt(n);i++)
    {
        if (n%i==0)
            return false;
    }
    return true;
}
```

完全平方数，百度了半天定义终于明白是个什么玩意了，说白了就是二次根号能开出整数的，64，121这种

```
bool PerfectSquare(int n)
{
    return pow((int)sqrt(n), 2) == n;
}
```
```c++
15.统计单词数（题干没有标点符号）
```c++
#include<iostream>
```
```

```
#include<stdlib.h>
using namespace std;
int main()
{
    FILE *fp;
    int num = 0;
    char c=NULL;
    if ((fp = fopen("D:\\in.txt", "r")) == NULL)
    {
        printf("Open error\n");
        exit(0);
    }
    c = fgetc(fp);
    while (c != EOF)
    {
        cout<<c;
        c = fgetc(fp);
        if (c == ' ')
            num++;
    }
    cout <<endl<< num+1 << endl;
    fclose(fp);
    return 0;
}
```

16.输入一行文本，A开头的单词和N结尾的单词互换位置，并输出

考试时间就2小时且不要求算法的空间和时间复杂度，所以怎么暴力怎么来，Node存单词位置和单词字母

```
#include<iostream>
#include<stdlib.h>
#include<stdio.h>
#include<string>
using namespace std;
struct Node
{
    int index;
    char word[50];
};
int main()
{
    freopen("d://in.txt", "r", stdin);
    char word[50];
    Node nodes[1000];
    int wordCount = 0;

    while (scanf("%s", &word) != EOF)
    {
        nodes[wordCount].index = wordCount;
        strcpy(nodes[wordCount].word, word);
        wordCount++;
    }
    for (int i = 0;i <= wordCount;i++)
    {
        if (nodes[i].word[0] == 'a' || nodes[i].word[0] == 'A')
        {
```

```
            for (int j = wordCount;j >= 0;j--)
            {
                if (nodes[j].word[strlen(nodes[j].word) - 1] == 'n' ||
                    nodes[j].word[strlen(nodes[j].word) - 1] == 'N')
                {
                    nodes[i].index = j;
                    nodes[j].index = i;
                    break;
                }
            }
        }
    }

    for (int i = 0;i <= wordCount;i++)
    {
        for (int j = 0;j <= wordCount;j++)
        {
            if (nodes[j].index == i)
            {
                cout << nodes[j].word << ' ';
                break;
            }
        }
    }
    cout << endl;
    return 0;
}
```

17.矩阵加法

矩阵加法就是对应位置上的数相加，两矩阵都为m*n

a[i][j]+=b[i][j];

矩阵乘法 M*N矩阵 和 N*Q矩阵

```
#include<iostream>
#include<stdlib.h>
using namespace std;
#define M 3
#define N 2
#define Q 4
int main()
{
    int a[M][N] = { 2,-6,3,5,1,-1 };
    int b[N][Q] = { 4,-2,-4,-5,-7,-3,6,7};
    int c[M][Q];
    for (int i = 0;i < M;i++)
    {
        for (int j = 0;j < Q;j++)
        {
            int sum = 0;
            for (int k = 0;k < N;k++)
            {
                sum += a[i][k] * b[k][j];
            }
            c[i][j] = sum;
        }
```

```
    }
    return 0;
}
```

18.验证哥德巴赫猜想

试试递归查找，默认情况下递归深度在4600左右程序崩溃，这里有做质数表的优化

```
#include<iostream>
#include<stdlib.h>
using namespace std;
int n, Prime[10000], maxIndex = 0;
void Check(int index1,int index2)
{
    if (index2 > maxIndex)
    {
        Check(index1 + 1, index1 + 1);
    }
    else if (n == Prime[index1] + Prime[index2])
    {
        cout <<n<< "=" << Prime[index1] << "+" << Prime[index2]<<endl;
        return;
    }
    else
    {
        Check(index1, index2 + 1);
    }
}
int main()
{
    memset(Prime, 0, sizeof(Prime));
    Prime[0] = 2;
    while (cin >> n)
    {
        if (Prime[maxIndex] > n)
        {
            Check(0, 0);
        }
        else
        {
            for (int i = Prime[maxIndex]+1;i <= n;i++)
            {
                bool flag = true;
                for (int j = 2;j <=sqrt(i);j++)
                {
                    if (i%j == 0)
                    {
                        flag = false;break;
                    }
                }
                if (flag)
                {
                    Prime[++maxIndex] = i;
                }
            }
            Check(0, 0);
        }
```

```
    }
    return 0;
}
```

循环查找版

就改个Check，数一大了计算就很慢了

```
void Check()
{
    for (int i = 0;i <= maxIndex;i++)
        for (int j = 0;j <= maxIndex;j++)
            if (Prime[i] + Prime[j] == n)
            {
                cout << n << "=" << Prime[i] << "+" << Prime[j] << endl;
                return;
            }
}
```

19.梯形法求定积分

以y=x^2为例，思路很简单，就是把不规则曲线所包围的图形分成一个个矩形

```
#include<iostream>
#include<stdlib.h>
using namespace std;
float f(float x)  //以x^2为例
{
    return x*x;
}
int main()
{
    int a, b;//上限和下限
    while (cin >> a >> b)
    {
        float w = (b - a) / 1000.0;//1000是精度,w是矩形宽
        float ans = 0;
        for (float i = a;i <= b;i += w)
        {
            ans += f(i)*w;  //f(i)是高
        }
        cout << ans <<endl;
    }
    return 0;
}
```

20.8皇后问题，题目很经典，题干不再赘述

二维数组版，思路就是以递归深度为查找的行数，定完行数循环猜列数，猜完后判断这个位置是否合理，然后递归下一行。递归到头后程序自己会return回去继续探索。

```
#include<iostream>
#include<stdio.h>
using namespace std;
int map[8][8];
int cnt = 0;
```

```
void Search(int row)
{
    if (row == 8)
    {
        cnt++;
        /*for (int i = 0;i < 8;i++)
        {
            for (int j = 0;j < 8;j++)
                printf("%2d", map[i][j]);
            putchar('\n');
        }
        putchar('\n');*/
    }
    else
    {
        for (int col = 0;col < 8;col++)
        {
            bool flag = true;
            for(int i=0;i<row;i++)//往上查
                if (map[i][col] == 1)
                {
                    flag = false;break;
                }
            for(int i=row-1,j=col-1;flag&&i>=0&&j>=0;i--,j--) //往左上查
                if (map[i][j] == 1)
                {
                    flag = false;break;
                }
            for(int i=row-1,j=col+1;flag&&i>=0&&j<=7;i--,j++)//往右上查
                if (map[i][j] == 1)
                {
                    flag = false;break;
                }
            if (flag)
            {
                map[row][col] = 1;
                Search(row + 1);
                map[row][col] = 0;
            }
        }
    }
}
int main()
{
    memset(map, 0, sizeof(map));
    Search(0);
    cout << cnt << endl;
    return 0;
}
```

一维数组版，跟二维数组思路一样，只是用一维数组的值代表列，下标代表行，判断上下斜方向的时候用了些数字技巧，能看懂的就看懂吧，反正我忘了

```
#include<iostream>
#include<stdio.h>
using namespace std;
int map[8];
```

```
int cnt = 0;
void Search(int row)
{
    if (row == 8)cnt++;
    else
    {
        for (int col = 0;col < 8;col++)
        {
            bool flag = true;
            for (int i = 0;i < row;i++)
            {
                if (map[i] == col//第i行的列是否等于本列
                    || map[i] - i == col - row
                    || map[i] + i == col + row)
                {
                    cout << map[i] - i << " " << col - row <<endl;
                    flag = false;break;
                }
            }
            if (flag)
            {
                map[row] = col;
                Search(row + 1);
            }
        }
    }
}
int main()
{
    memset(map, 0, sizeof(map));
    Search(0);
    cout << cnt << endl;
    return 0;
}
```

21.字符串倒置

```
#include<iostream>
#include<stdio.h>
using namespace std;
char* strReverse(char *str)
{
    int len = strlen(str);
    for (int i = 0;i < len / 2;i++)
    {
        /*char c = str[i];
        str[i] = str[len - i - 1];
        str[len - i - 1] = c;*/
        swap(str[i], str[len - i - 1]);
    }
    return str;
}
int main()
{
    char str[] = "MaxLykoS";
    cout << strReverse(str);
    return 0;
```

```
}
```

22.快速排序，思路很直接，找到一个标杆，比如n[0]，然后先从右往左遍历，找到一个比标杆小的，移动到左边，再从左往右遍历，找到一个比标杆大的，移动到右边，最后左下标和右下标会撞到一起（在中间），再把标杆放到中间。不用担心奇偶。一轮交换后，就会是标杆在中间，大于标杆的在一边，小于的在一边，然后递归左边的数们和右边的数们。

这里描述的其实有些跳跃，所谓的“移动”，并不是说两值交换，而是覆盖赋值，也就是说，每一次移动，数组中都会有两个一模一样的数，但是别忘了标杆是单独存储的。所以当两下标相撞时，中间会有两个紧挨着的相等的数，再把标杆赋给中间值，就不会有重复的数了。换句话说，就是标杆存起来，当作两值交换用的那个t，但只出现在交换的最开始和最末尾，再换句话说，就是大型连锁交换现场（笑）

```
#include<iostream>
#include<stdio.h>
using namespace std;
int n[] = {  57, 68, 59, 52, 72, 28, 96, 33, 24 };
void Qsort(int low, int high)
{
    if (low >= high)return;
    int first = low, last = high;
    int key = n[low];
    while (first < last)
    {
        while (first < last&&n[last] <= key)last--;
        n[first] = n[last];
        while (first < last&&n[first] >= key)first++;
        n[last] = n[first];
    }
    n[last] = key;
    Qsort(low, first - 1);
    Qsort(last + 1, high);
}
int main()
{
    Qsort(0, 8);
    for (int i = 0;i < 9;i++)
        cout << n[i] << " ";
    cout << endl;
    return 0;
}
```

23.三阶魔方阵

```
#include<stdio.h>
#include<iostream>
using namespace std;
bool book[10];//1 2 3
int mark[10]; //4 5 6
int cnt = 0;  //7 8 9
void Search(int deepth)
{
    if (deepth == 10)
    {
        int sum = mark[1] + mark[2] + mark[3];
        if (mark[4] + mark[5] + mark[6] == sum
```

```
            &&mark[7] + mark[8] + mark[9] == sum
            &&mark[1] + mark[4] + mark[7] == sum
            &&mark[2] + mark[5] + mark[8] == sum
            &&mark[3] + mark[6] + mark[9] == sum
            &&mark[1]+mark[5]+mark[9]==sum
            &&mark[3]+mark[5]+mark[7]==sum)
        {
            cnt++;
        }
    }
    else
    {
        for (int i = 1;i <= 9;i++)
        {
            if (book[i] == false)
            {
                book[i] = true;
                mark[deepth] = i;
                Search(deepth + 1);
                book[i] = false;
            }
        }
    }
}
int main()
{
    memset(book, 0, sizeof(mark));
    memset(mark, 0, sizeof(mark));
    Search(1);
    cout << cnt << endl;
    return 0;
}
```

24.一些计算机二级c语言题目的坑

3<sqrt(12) 成立，别一看跟int比就把sqrt返回值脑补为int

按 .2%f类似格式输出的时候小数点有四舍五入

0xF0 二进制为1111 0000，十进制为240

for (i = 0;str[i] != '\0';i++);求字符串长度的时候别忘了最后的\0

#define SUM(a,b) a+b这不是个函数，比如SUM(5,9)/2在程序里就是5+9/2，答案是9不是7

\065是八进制表示53，字符为'a'

25.

一只刚出生的奶牛，4年生1只奶牛，以后每一年生1只。现在给你一只刚出生的奶牛，求20年后有多少奶牛，假设奶牛是个寿仙。模拟

```
#include<stdio.h>
#include<iostream>
using namespace std;
int main()
{
    int cows[5] = {1,0,0,0,0};
    for (int i = 0;i < 20;i++)
```

```
    {
        int newBorn = cows[3] + cows[4];
        for (int j = 3;j >=0;j--)
        {
            if(cows[j]>0)
            {
                cows[j + 1] += cows[j];
                cows[j] = 0;
            }
        }
        cows[0] += newBorn;
    }
    cout << cows[0] + cows[1] + cows[2] + cows[3]+ cows[4] << endl;
    return 0;
}
```