Week3 算法作业

1. 求 1×2×3…×n 的末尾有多少个 0？n 由键盘输入，且 1000<n<10000。

解答：高精度乘法，计算出结果之后统计出末尾 0 的个数，即为答案

代码：

```
// 大整数乘法
#include <cstring>
#include <iostream>

using namespace std;

const int MAXN = 1000;
int A[MAXN], B[MAXN], C[MAXN];  // 大整数容器
int len_a, len_b;
char b[MAXN];
int n;

// 将 A 和 B 相加，结果仍然存入 A
void mul() {
    memset(C, 0, sizeof(C));
    for (int i = 0; i < len_a; i++) {
        for (int j = 0; j < len_b; j++) {
            C[i + j] += A[i] * B[j];
            C[i + j + 1] = C[i + j] / 10;
            C[i + j] = C[i + j] % 10;
        }
    }

    for (int i = 0; i < MAXN; i++) {
        A[i] = C[i];
    }

    len_a = MAXN;
    while (A[len_a - 1] == 0) len_a--;
}

// 将 b 逐位分解存下来
void set_b(int t) {
    int count = 0;
    while (t > 0) {
        B[count++] = t % 10;
        t = t / 10;
    }
    len_b = count;
```

```
}
void init(){
    cin >> n;
    A[0] = 1;
    len_a = 1;
}
void print_ans(){
    int k = MAXN-1;
    while(A[k]==0) k--;
    cout <<"fac("<<n<<") = ";
    for (; k >= 0; k--) {
        cout<< A[k];
    }
    cout<<endl;
    k = 0;
    while(A[k]==0)k++;
    cout<<"Total zero: "<<k<<endl;
}
int main() {
    init();
    for(int i=1;i<=n;i++){
        set_b(i);
        mul();
    }
    print_ans();
    return 0;
}
```

运行结果 1：

```
10
fac(10) = 3628800
Total zero: 2
```

运行结果 2：

```
50
fac(50) = 30414093201713378043612608166064768844377641568960512000000000000
Total zero: 12
```

2. 构造 m 行 n 列的逆转矩阵，例如当 m=4，n=5 时：

1 14 13 12 11
2 15 20 19 10
3 16 17 18 9
4 5 6 7 8

解答：模拟步进填数，精确控制每圈的填数长度和宽度，逐圈向中心逼近，最终完成填数。

代码：

```cpp
// 蛇形填数
#include <cstring>
#include <iostream>
using namespace std;
const int MAXN = 100;

int matrix[MAXN][MAXN];

int r, c;

void print_matrix(int m, int n) {
    for (int i = 1; i <= m; i++) {
        for (int j = 1; j <= n; j++) {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
    cout << endl;
}

void solve(int m, int n) {
    memset(matrix, 0, sizeof(matrix));
    r = 0;
    c = 1;
    int count = 1;
    while (m > 0 && n > 0) {
        for (int i = 0; i < m; i++) {
            matrix[++r][c] = count++;
        }
        for (int i = 0; i < n - 1; i++) {
            matrix[r][++c] = count++;
        }
        if (m <= 1 || n <= 1) break;
        for (int i = 0; i < m - 1; i++) {
            matrix[--r][c] = count++;
        }
        for (int i = 0; i < n - 2; i++) {
            matrix[r][--c] = count++;
        }
        m = m - 2;
        n = n - 2;
    }
}
```

```
int main() {
    int m, n;
    cin >> m >> n;
    solve(m, n);
    print_matrix(m, n);
    return 0;
}
```

运行结果 1：

```
5 5
1 16 15 14 13
2 17 24 23 12
3 18 25 22 11
4 19 20 21 10
5 6 7 8 9
```

运行结果 2：

```
10 15
1 46 45 44 43 42 41 40 39 38 37 36 35 34 33
2 47 84 83 82 81 80 79 78 77 76 75 74 73 32
3 48 85 114 113 112 111 110 109 108 107 106 105 72 31
4 49 86 115 136 135 134 133 132 131 130 129 104 71 30
5 50 87 116 137 150 149 148 147 146 145 128 103 70 29
6 51 88 117 138 139 140 141 142 143 144 127 102 69 28
7 52 89 118 119 120 121 122 123 124 125 126 101 68 27
8 53 90 91 92 93 94 95 96 97 98 99 100 67 26
9 54 55 56 57 58 59 60 61 62 63 64 65 66 25
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
```

3. 两个球队进行比赛，各选 3 人，甲队为 A、B、C 三人，乙队为 X、Y、Z 三人，已经抽签决定比赛名单。现在知道 A 不和 X 比，C 不和 X、Z 比，请寻找出三对选手的比赛名单。

解答：设甲队出场顺序为 A-B-C，利用 algorithm 中的 next_permutation 函数枚举 X、Y、Z 三人的出场顺序，同时检验每个顺序是否满足条件。

代码：

```
// 赛手名单
// 以 A,B,C 为序，寻找对应的 X,Y,Z 顺序，使其满足所有条件
#include <algorithm>
#include <iostream>

using namespace std;
int main() {
    int order[] = {0, 1, 2};
    char name[] = "XYZ";
    bool flag = false;
    do {
        if (order[0] != 0 && order[2] != 0 &&
```

```
            order[2] != 2) {  // A 不和 X 比，C 不和 X、Z 比
            flag = true;
            break;
        }
    } while (next_permutation(order, order + 3));
    if (flag) {
        cout << "A-" << name[order[0]] << "\n";
        cout << "B-" << name[order[1]] << "\n";
        cout << "C-" << name[order[2]] << "\n";
    }
    else {
        cout<<"No proper arrangement found\n";
    }
    return 0;
}
```

运行结果：

```
A-Z
B-X
C-Y
```

4. 求 2+22+222+……+22..22（最后一个数共 n 个 2，不考虑精度）

解答：利用循环直接加和即可：

代码：

```
// 求 2+22+222+……+22..22

#include<iostream>
#include<algorithm>

using namespace std;

typedef long long ll;

void solve(int n){
    ll k = 2;
    ll sum = 0;
    for(int i=0;i<n;i++){
        sum+=k;
        k = k*10+2;
    }
    cout<<sum<<endl;
}

int main(){
```

```
    int n;
    cin>>n;
    solve(n);
    return 0;
}
```

运行结果 1：

```
5
24690
```

运行结果 2：

```
10
2469135800
```

5. 设 N 是一个正整数，求比 N 大的最小“不重复数”，这里的不重复指的是没有两个相邻的数字相等。

解答：从最高位向右检索，找到第一个重复的相邻两位，将次高位+1，低位设置为 0101…重复上述过程直到构造出不重复数为止，

```
// 最小不重复数
//
#include <cmath>
#include <iostream>
using namespace std;

//求解数字的位数
int numLength(int N) {
    int len = 0;
    while (N > 0) {
        N = N / 10;
        len++;
    }
    return len;
}
//求解最小不重复数
int minNoneDNum(int N) {
    bool isDNum = false;  // 是重复数的标志
    N++;                  // 自增 1，确保比原来的数大
    do {
        isDNum = false;
        int len = numLength(N);  // 从最高位向右遍历，检索是否有重复的两位
        int a, b;                // 最高位和次高位
        for (int i = len - 1; i >= 1; i--) {
            int d = pow(10, i);
            a = N / d % 10;
            b = N / (d / 10) % 10;
            if (a == b) {  // 发现重复的两位,在次高位+随后在后面补 01
                isDNum = true;
```

```
                N = N / (d / 10) + 1;
                int temp = 0, k = i-1;
                while (k > 0) {  // 低位置为 0 和 1
                    N = N * 10 + temp;
                    temp = 1 - temp;
                    k--;
                }
                cout <<"->"<< N << endl;
            }
        }
    } while (isDNum);
    return N;
}
int main() {
    int N;
    cin>>N;
    int ans = minNoneDNum(N);
    cout<<"Answer:"<<ans<<endl;
    return 0;
}
```

运行结果 1:

```
989899
->990001
->990010
->1000101
->1001010
->1010101
Answer:1010101
```

运行结果 2：

```
1111
->1201
Answer:1201
```

6. 两个数组 a[N]，b[N]，其中 a[N]的各个元素已知，现在给 b[n]赋值，其中

$$\boldsymbol{b[i] = a[0] \times \ldots \times a[N-1]/a[i]}$$

要求：

1. 不使用除法运算
2. 除了循环计数值、a[N]、b[N]以外，不使用其他任何变量
3. 满足时间复杂度 O(N)，空间复杂度 O(1)

解答：

参照课上思路，依照下面三步进行赋值：

1. 令$b[i] = a[i+1] \times \ldots \times a[N-1],\ i = N-2, N-1 \ldots 0$
2. 令$a[i] = a[0] * \ldots * a[i],\ i = 1 \ldots n-1$
3. 接下来遍历 b 数组进行赋值：$b[i] = a[i-1] \times b[i],\ i = 1,2 \ldots N-1$

代码：

```cpp
// 2012 百度面试题
#include<iostream>
using namespace std;

#define N 10
int a[N] = {1,2,3,4,5,6,7,8,9,10},b[N];

void solve(){
    b[N-1] = 1;
    for(int i=N-2;i>=0;i--) b[i] = b[i+1]*a[i+1];
    for(int i=1;i<N;i++) a[i] = a[i]*a[i-1];
    for(int i=1;i<N;i++) b[i] = a[i-1]*b[i];
}


int main(){
    cout<<"a[N]: ";
    for(int i=0;i<N;i++){
        cout<<a[i]<<" ";
    }
    solve();
    // 输出结果
    cout<<endl;
    cout<<"b[N]: ";
    for(int i=0;i<N;i++){
        cout<<b[i]<<" ";
    }
    cout<<endl;
    return 0;
}
```

运行结果：

```
a[N]: 1 2 3 4 5 6 7 8 9 10
b[N]: 3628800 1814400 1209600 907200 725760 604800 518400 453600 403200 362880
```