**Анализ на Задача Cx. Брой низове**

Да означим с *S* множеството от всички низове с дължина 2*n*, образувани от първите *n* главни латински букви и съответните им *n* малки латински букви и такива, че всяка главна буква е разположена преди съответната ѝ малка буква.

За стойности на *n* ≤ 6 могат да се генерират всички низове от множеството *S* и да се преброят.

До известно улеснение и опростяване на програмата води следното наблюдение.

Ако в низовете от множеството *S*  заменим всяка главна буква със съответната ѝ малка буква, ще получим друго множество от низове *T*. Низовете от множеството *T* са образувани от 2*n*  букви {a,a,b,b,c,c,…}. Обратно, ако в низ от множеството *T* заменим първата от двете малки букви със съответната голяма буква, ще получим низ от множеството *S.* Следователно вместо да генерираме низовете от множеството *S*,може да генерираме низовете от множеството *T*. Освен това, няма никакво значение дали ще използваме букви a, b, c, … или числа 1, 2, 3, ….

 vector<int> a;

 for(int i=1; i<=n; i++)

 { a.push\_back(i);

 a.push\_back(i);

 }

 int t=0;

 do

 { t++;

 } while(next\_permutation(a.begin(),a.end()));

 cout << t << endl;

Този подход решава задачата за *n* ≤ 6.

За по-големи стойности на *n* трябва да използваме формула (или рекурентна формула) за пресмятане на *аn*, където с *аn* е означен броят на елементите в множеството *S* (или *T*) за дадено *n.* Броят на пермутациите на 2*n* **различни** елемента е (2*n*)!. За всяка двойка от равни елементи трябва да разделим на 2.

Така получаваме формулата *аn* = (2*n*)! / 2*n*.

Например при *n* = 8 получаваме

*а8* = 16! / 2*8* = (16.15/2).(14.13/2)(12.11/2)...(4.3/2).(2.1/2)

Следният фрагмент от програма реализира съответните пресмятания:

 long long a=1;

 for(int k=2; k<=2\*n; k=k+2)

 a = a \* k\*(k-1)/2;

 cout << a << endl;

 Тази подход решава задачата за *n* ≤ 11, защото за по-големи стойности на *n* резултатът не се събира в променлива от тип **long long**.

 За по-големи стойности на *n* трябва да се реализира аритметика с „дълги“ числа и по-конкретно умножение на дълго число с „едноцифрено“ число.

 Следва една възможна реализация.

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

vector<int> mul(vector<int> a,int x)

{ int n=a.size();

 vector<int> b(n);

 int c=0;

 for(int i=0; i<n; i++)

 { int p = a[i]\*x + c;

 b[i] = p%10;

 c = p/10;

 }

 if(c>0) b.push\_back(c);

 return b;

}

int main()

{ int n; cin >> n;

 vector<int> a({1});

 for(int k=2; k<=2\*n; k=k+2)

 a = mul(a,k\*(k-1)/2);

 for(int i=a.size()-1; i>=0; i--)

 cout << a[i];

 cout << endl;

 return 0;

}

*n* *an*

--------------------------------------------------------------

1 1

2 6

3 90

4 2520

5 113400

6 7484400

7 681080400

8 81729648000

9 12504636144000

10 2375880867360000

11 548828480360160000

12 151476660579404160000

13 49229914688306352000000

14 18608907752179801056000000

15 8094874872198213459360000000

16 4015057936610313875842560000000

17 2252447502438386084347676160000000

18 1419041926536183233139035980800000000

19 997586474354936812896742294502400000000

20 778117449996850714059458989711872000000000

21 669959124447288464805194190141921792000000000

22 633781331727134887705713703874258015232000000000

23 655963678337584608775413683509857045765120000000000

24 739927029164795438698666634999118747623055360000000000

25 906410610726874412405866627873920465838242816000000000000

26 1201900469823835470850179148560818537701509974016000000000000

 *Стоян Капралов*