**2.Възстановяване на скоби**

Задача: Даден е шаблон от кръгли скоби и въпросителни знаци.Трябва да се намери по колко начина е възможно въпросителните знаци да се заместят с кръгли скоби,така че да се получи пражилен израз в скоби.

\**Тук няма да се спираме върху точната дефиниция на понятието правилен израз от скоби.*

Пример : При даден шаблон **‘????(?’** отговорът на задачата е **2.**

Решение : да разгледаме една редица **s** от отварящи и затварящи кръгли скоби с общ брой на скобите(дължина на редица),равен на **n** .Означваме с **di(s)** разликата между броя на отварящите скоби и броя на затварящите скоби,които се намира измежду първите **i** знака(включително) в тази редица.Очевидно е ,че **s** е правилен израз от скоби тогава и само тогава,когато **di(s)>0** за всяко **i=1,2,…….,n** и освен това **dn(s)=0.**

Нека **F(k,c)** да означава броя на всевъзможните редици **s** от скоби с дължина **k** ,които се съгласуват с първите **k** знака на дадения шаблон и които са такива ,че **di(s)>0** за всяко **i=1,2,….,k** и **dk(s) = c.**Сега е ясно,че отговорът на първоначално поставената задача е равен на **F(N,0)**, където **N** е дължината на дадения шаблон.

Остава да немерим начина за пресмятане на числата **F(k,c)**.Забелязваме,че

- ако **k**-тият знак на шаблона е отваряща скоба,то

F(k,c) = F(k - 1,c - 1);

- ако **k**-тият знак на шаблона е затваряща скоба,то

F(k,c) = F(k - 1,c + 1);

* ako **k**-тият знак на шаблона е въпросителен знак,то на неговото място може да се постави както отваряща,така и затваряща скоба и следователно

F(k,c) = F(k – 1,c – 1)+F(k – 1,c + 1);

Лесно се съобразяват началните условия : F(0,c) = 0 , F(k, -1) = 0 и F(0,0) = 1 (празната редица).

Пресмятането се извършва чрез два вложени цикъла: външен – в който се променя индексът **k**, и вътрешен – в който се променя индексът **с**. Така последователно се пресмятат числата **F(k,c)** за всички **k** от **1** до **N** и за всички c от **0** до **N** . Стойността **F(N,0)** e отговор на задачата .

Една реализация на програма , запълваща масива F [k] [c] , e следната :

const N = 6;

char s [] = “ ????(?” ;

void main ( )

{ int F[N+1] [N+2];

int c, k;

for (c = 0; c <= N+1; c++) F [0] [c] = 0 ;

for (k = 0; k <= N; k++) F [k] [N+1] = 0;

F [0] [0] = 1;

for (k = 1; k <= N; k++) {

c = 0;

if (s[k] = = ` (`) F [k] [c] = 0;

else if (s[k] = = `)` ) F [k] [c] = F [k-1] [c+1];

else F [k] [c] = F [k-1] [c+1];

for (c = 1; c <= N; c++)

if (s [k] = = ` ( ` ) F[k] [c] = F [k-1] [c-1];

else if (s [k] = =`)` ) F [k] [c] = F [k-1] [c+1];

else F [k] [c] = F [k-1] [c-1] + F [k-1] [c+1];

}

Cout << F [N] [0] ;

}

Привеждаме и пример за рекурсивна програма , решаваща задачата:

const N = 6

char s[] = ” ????(?”;

int F ( int k, int c )

{ if ((k = = 0) &&(c = = 0)) return 1;

if (k = = 0) return 0 ;

if (c = = -1) return 0;

if (c = = N+1) return 0;

if (s[k] = = `(` ) return F (k-1, c-1);

else if ( s[k] = = `)` ) return F (k-1, c+1);

else return F(k-1, c-1) + F(k-1, c+1);

}

void main ( )

{ cout << F(N, 0);

}

**4. Ходове с коня**

Едно шахматно дружество поръчало на телефонната компания да осигури за шахматистите телефонни номера , които да се избират чрез ходове на шахматния кон бърху телефонната клавиатура.Стандартната клавиатура е изобразена по долу:

789

456

123

0

Пример за телефонен номер от описания вид е 340-49-27.

*Задача*. *Напишете програма, която намира броя на различните телефонни номера с дължина* ***N*** *, които да се набират с ходове на коня по описания начин. Номерата не могат за започват с* ***0*** *.*

*Упътване:* Нека **F(k,d)** да означава броя на телефонните номера от търсения вид с дължина (броя на цифрите на номера), равни на **k** , като първата цифра е **d.** Напишете рекурентни зависимости за числата **F(k,d)**. Например **F(k + 1,6) = F(k,0) = F(k,1) + F(k,7),** защото бутонът 6 може да се достигне с ход на коня от бутоните 0, 1 или **7.**

След като намерим **F(N,d)** за всяко **d= 1,2,…..,9**, отговорът на задачата е

**Образуване на суми**. *Разполагаме с по една пощенска марка от* 1, 2, 3, 4 *и* 5 *лева. Колко са начините за облепване на писмо, за да го таксуваме с* 10 *лева*?

За да решим задачата, потапяме я във фамилия от подзадачи, зависещи от два параметъра: S и k, където S e парична сума за таксуване, а k показва колко от първите няколко марки ще използваме, за да образуваме сумата. По-общо можем да означим стойностите на марките с a1, a2, …, aN. Нека F(k, S) е броят на начините за таксуване при фиксирано k и S.

Разделяме всички начини за таксуване на два вида – със и без използване на марката със стойност ak. Ако е използвана, то остава да се образува сумата S – ak чрез останалите марки a1, …, a k–1; ако не е използвана, тогава със същите марки трябва да се образува сумата S. Първата възможност се осъществява по F(k–1, S–ak) начина, а втората – по F(k–1, S). Тогава общият брой на възможностите е равен на сбора от двете подзадачи:

F(k, S) = F(k–1, S–ak) + F(k–1, S).

Тази формула, заедно с очевидни начални условия, води естествено към рекурсивно програмиране. Но не е трудно да избегнем експоненциалния ръст при рекурсията чрез използване на таблична техника.

**2. Зад. Броене**

Густаво знае да брои, но сега той се учи да пише числата. Като много добър ученик той е научил 1, 2, 3 и 4. Но той не осъзнава, че 4 е различно от 1. Въпреки това той се забавлява с една игра, в която съставя числа от тези цифри и смята сбора на цифрите. Например :

132 = 1 + 3 + 2 = 6

112314 = 1 + 1 + 2 + 3 + 1 + 1 = 9 ( запомнете, че Густаво мисли, че 4 = 1)

След като създава много числа по този начин, сега той иска да разбере колко числа може да направи така, че тяхната сума да е равна на N. Например за N = 2 той може да напише 5 числа: 11, 14, 41, 44 и 2. Понеже бил зает с учене, той Ви моли за помощ.

Входните данни се четат от стандартния вход, в който има само едно цяло число N.(1 <= N <= 1000).

Резултата трябва да бъде изведен на стандартния изход.

**Пример:**

Вход: Изход:

1. 13

**Анализ на задачата:** Задачата като разсъждения много прилича на първата. Само, че тук не разглеждаме дължина, а сума. Ще мислим от гледна точка на добавяне на единица към сумата. Сумата n може да се получи или като n-1 + 1, n-2 + 2, n-3 + 3, като първия вариант трябва да го преброим два пъти, тъй като вместо 1 може да се напише 4.

**Описание на алгоритъма:** Алгоритъма е изключително тривиален – цялата работа се свършва в един цикъл с три предварителни присвоявания.

**Описание на използваните променливи и структури от данни:** Използваме един масив, в който пазим бройките – b[].

**Реализация на задачата в езика за програмиране C++ :**

#include <iostream>

using namespace std;

int n, b[1024];

int main()

{

cin >> n;

b[1] = 2;

b[2] = 5;

b[3] = 13;

for (int i = 4; i <= n; i++)

b[i] = 2\*b[i-1] + b[i-2] + b[i-3];

cout << b[n] << endl;

return 0;

}

**3. Образуване на суми**

*Задача.Разполагаме с по една пощенска марка от 1, 2, 3, 4 и 5 лева. Колко са начините за облепване на писмо, за да го таксуваме с 10 лева?*

За да решим задачата, „потапяме” я във фамилия от подзадачи, зависещи от два параметъра : **S** и **k**, където **S**  е парична сума за таксуване, a **k** показва колко от първите няколко марки ще използваме,за да образуваме сумата. По-общо можем да означим стойностите на марките с **а1, а2, .... , аN**. Нека **F(k,S)** е броят на начините за таксуване при фиксирано **k** и **S.**

Разделяме всички начини за таксуване на два вида – със и без използване на марката със стойност **аk**. Ако е използвана, то остава да се образува сумата **S- ak** чрез останалите марки **а1, а2, .... , аk-1**; ако не е използвана, тогава със същите марки трябва да се образува сумата **S**. Първата възможност се осъществява по F(k-1, S-ak) начина,а втората по **F(k-1, S)**. Тогава общият брой на възможностите е равен на сбора от двете подзадачи:**F(k,S)=F(k-1,S-ak)+F(k-1,S).**

Тази формула,заедно с очевидно начални условия,води естествено към рекурсивно програмиране (стойностите **а1,....,аN** са заредени в елементите

**а [1] ,….a [N]** ):

Int F (int k, int S)

{ if (S<0) return 0;

if(S = = 0) return 1;

if (k = = 1)

if (a [1] = = S) return 1;

else return 0;

return F(k-1, S-a [k] ) + F(k-1, S);

}

Не е трудно да изберем експоненциалния ръст при рекурсията чрез използване на таблична техника.Следващият програмен фрагмент отпечатва числото 3, което възможният брой начини за образуване на сумата от задачата:

const S = 10;

const N = 5;

int a [N+1] = {0,1,2,3,4,5};

int T [N+1] [S+1], k, j, w;

for (j = 0; j <= S; j++) T [0] [j] = 0;

for (k = 1; k <= N; k++) T[k] [0] = 1;

for (k = 1; k <= N; k++)

if (a [1] = = 1) T [k] [1] = 1;

else T [k] [1] = 0;

for (k = 1; k <= N; k++)

for (j = 2; j <= S; j++) {

if (j-a [k] < 0) w = 0;

else w = T [k-1] [j-a [k]];

T [k] [j] = w + T [k-1] [j];

}

cout << T [M] [S];