РЕКУРСИЯ (*допълнителни задачи*)

Зад. 1. Да се напише рекурсивна функция int min(size_t n, int a[]), която намира минималния елемент на вектора $a = (a_1, a_2, \ldots, a_n)$.

Зад. 2. Да се напише рекурсивна функция int sum(size_t n, int a[]), която намира сумата от елементите на вектора $a = (a_1, a_2, \ldots, a_n)$.

Зад. 3. Да се напише рекурсивна функция, която по дадено естествено число n пресмята произведението на естествените числа от 1 до n със стъпка 3, т.е. $P = 1.4.7.10...n$.

Зад. 4. Да се напишат итеративна и рекурсивна функция за изчисляване стойността на полинома на *Hermite* (https://en.wikipedia.org/wiki/Hermite_polynomials)

(x е реална, а n е цяла променлива $n \geq 0$), дефиниран по следния начин:

$$\begin{cases} h_0(x) = 1 \\ h_1(x) = 2x \\ h_n(x) = 2xh_{n-1}(x) - 2(n-1)h_{n-2}(x),\ n > 1. \end{cases}$$

Зад. 5. Да се напише програма, която въвежда редица от символи с произволна дължина. Въвеждането да продължи до въвеждане на символа '!'. Програмата да извежда редицата от символи в обратен ред (Да не се използва масив).

Пример:

вход	изход
recursion!	!noisrucer

Зад. 6. Дадена е матрица $N \times N$. Да се напише рекурсивна програма, която развива матрицата A по спирала в посока по часовниковата стрелка.

Например ако за $N = 3$, A има вида $A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$

да се конструира редицата $B = (a_{11}, a_{12}, a_{13}, a_{23}, a_{33}, a_{32}, a_{31}, a_{21}, a_{22})$.

Анализ на задачата:

Ще използваме рекурсивната функция void spiral(), която развива по спирала частта A' на A

$$A' = \begin{pmatrix} a_{k,k} & a_{k,k+1} & \cdots & a_{k,n-k+1} \\ a_{k+1,k} & a_{k+1,k+1} & \cdots & a_{k+1,n-k+1} \\ \cdots & & & \\ a_{n-k+1,k} & a_{n-k+1,k+1} & \cdots & a_{n-k+1,n-k+1} \end{pmatrix}$$

Зад. 7*. Дадена е двумерна мрежа от клетки (матрица), всяка от които е празна или запълнена. Запълнените клетки, които са свързани, т.е. имат съседни в хоризонтално,

вертикално или диагонално направление, образуват област. Да се напише програма, която намира броя на областите и размера (в брой клетки) на всяка област.

Анализ на задачата:

Ще дефинираме функция int countAreas(), която преброява клетките в областта, съдържаща дадена клетка (X, Y). Функцията има два параметъра X и Y - координатите на точката и реализира следния алгоритъм:

a) Ако клетката с координати X и Y е извън мрежата, приемаме, че броят е равен на 0.
b) Ако клетката с координати X и Y е празна, приемаме, че броят е равен на 0.
c) В останалите случаи, броят на клетките от областта е равен на сумата от 1 и броя на клетките на всяка област, на която принадлежат осемте съседни клетки на клетката с координати (X, Y).

От подточка c) следва, че функцията int count() е рекурсивна. За да избегнем зацикляне, трябва преди рекурсивното обръщение да направим клетката (X, Y) празна.

Зад. 8.** Дадени са n града (n е естествено число $n \geq 1$) и целочислена матрица $A_{n \times n}$, така че:

$$A_{i,j} = \begin{cases} 1, ako \ominus \\ 0, ako \oslash \end{cases}$$

$\ominus$ - има пряк път от i-тия до j-тия град;

$\oslash$ - няма пряк път от i-тия до j-тия град.

$(0 \leq i \leq n, 0 \leq j \leq n)$. Да се напише програма, която установява дали съществува път между два произволно зададени града. (Приемаме, че ако от град i до град j има път, то има път от град j до град iсъщо).

Да се изведе намерения път.

Анализ на задачата:

Ще напишем рекурсивна булева функция way, която зависи от два параметъра i и j, показващи номерата на градовете между които търсим дали съществува път. Функцията ще реализира следния алгоритъм:

- Ако $i = j$, съществува път от град i до град j.
- Ако i или j са извън матрицата на съседства, няма път.
- Ако $i \neq j$ и има пряк път от град iдо град j, има път между двата града.
- В останалите случаи има път от град i до град j, тогава и само тогава, когато съществува град k, с който град i е свързан с пряк път, и от който до град j има път.

Пътят се записва в едномерен масив int path[100], а дължината му се записва в променлива s.