10. Funkcje rekurencyjne

1 Problem 8 hetmanów

Problem ośmiu hetmanów – problem polegający na wyznaczeniu liczby różnych rozmieszczeń ośmiu hetmanów na szachownicy $n \times n$ tak, aby wzajemnie się nie atakowały.

Niech (i, j) i (m, n) będą współrzędnymi dwóch hetmanów. Dwa hetmany się atakują, jeżeli

- stoją na jednej linii, czyli $i = m$ lub $j = n$
- stoją na jednej przekątnej, czyli $i = m \pm x$, $j = n \pm x$ gdzie znak x może być identyczny bądź różny w obydwu równaniach

Szablon programu należy uzupełnić o definicję następujących funkcji

1. `int place_queens(int* queens, const int n, const int ndx)` – funkcja rekurencyjna; `queens` – tablica pozycji hetmanów, `n` – rozmiar szachownicy, `ndx` – liczba hetmanów już ustawionych.
2. `void print_board(const int* queens, const int n)` – wypisz pozycje hetmanów

Ponieważ z warunków zadania wynika, że w każdym rzędzie szachownicy znajduje się dokładnie jeden hetman, ich pozycje możemy zapisać w tablicy jednowymiarowej o rozmiarze `n`; `queens[i] == j` oznacza, że hetman znajduje się na pozycji `[i][j]`.

Program powinien wypisać k-te rozwiązanie (zaczynając od $k = 1$) przy założeniu, że wszystkie rozwiązania są posortowane rosnąco wg kolejnych liczb w tablicy `queens` (najpierw porównujemy wartości w `queens[0]`, w przypadku równych wartości porównujemy `queens[1]`, itd.).

Jeżeli k jest większe niż maksymalna liczba rozwiązań, program wypisuje wartość -1.

- **Wejście**

  2 liczby całkowite: $2 \leqslant n \leqslant 15$ – rozmiar szachownicy, $k \geqslant 1$ – numer rozwiązania jakie należy wypisać.

- **Wyjście**

  Program wyprowadza dokładnie n liczb całkowitych, stanowiących k-te rozwiązanie zadania. Jeżeli dla danego n jest mniej niż k rozwiązań, program powinien wypisać jedną wartość: -1.

- **Przykład**

  Wejście:

  ```
  5 1
  ```

  Wyjście:

  ```
  0 2 4 1 3
  ```