**Доклад**

**по**

**Алгоритми и структури от данни**

**Тема: Динамично програмиране и приложение на рекурсията**

1. **Увод в динамичното програмиране**
2. **Описание на задачи и решения с компилиран код**
3. **Двумерни задачи за динамично програмиране. Примери**
4. **Приложение на рекурсията в динамичното програмиране. Примери**
5. **Източници**

Изготвил: Ваня Ванева 12а

1. **Увод в динамичното програмиране**

Разбиване на задача на подзадачи: Динамичното програмиране е метод за решаване на сложни задачи, като ги разбива на по-малки и управляеми подзадачи.

Състояние: В динамичното програмиране въвеждаме понятието за състояние, което описва текущото състояние на проблема.

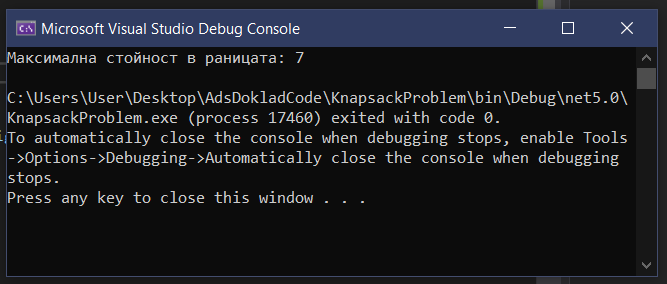
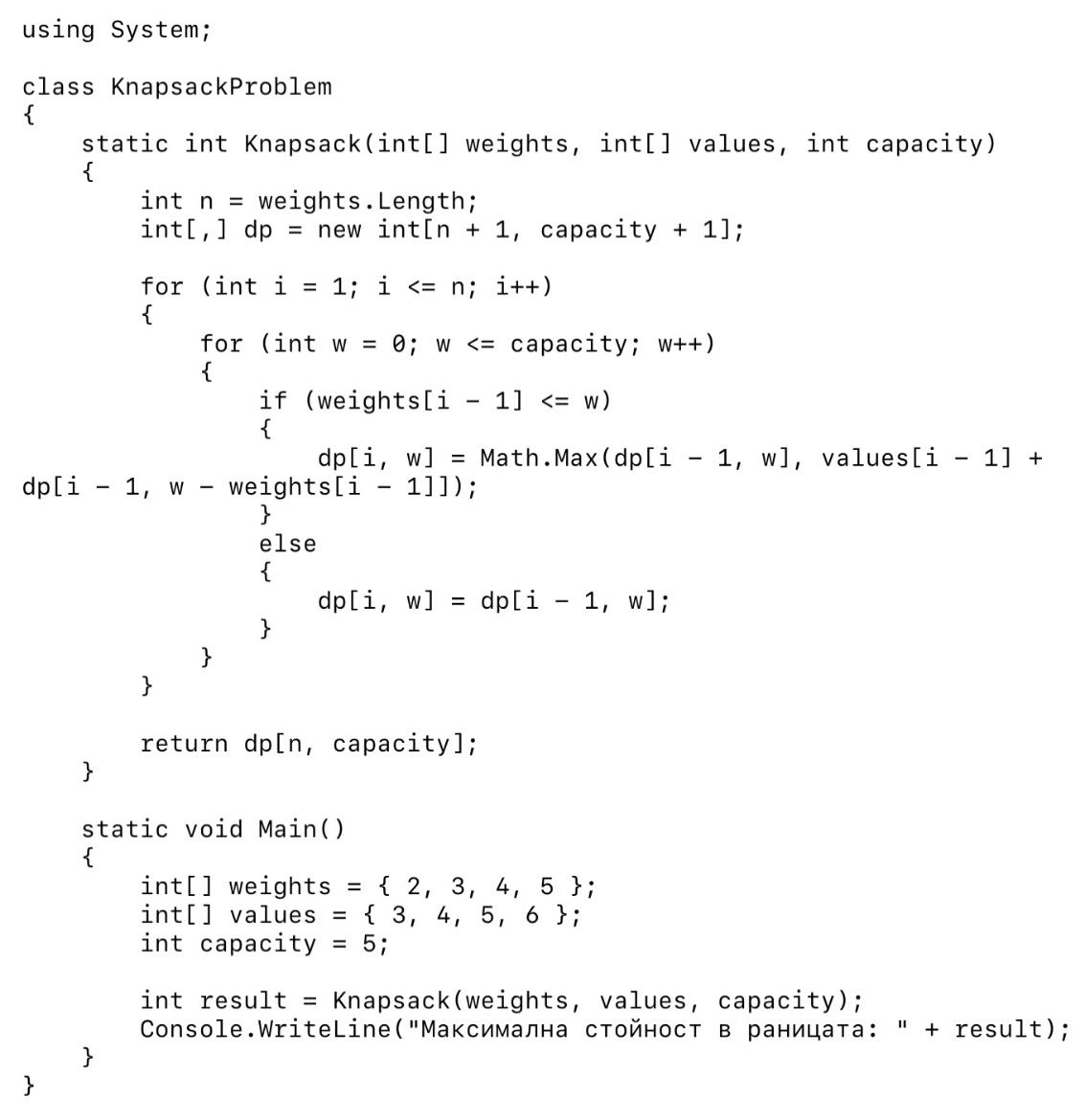
Мемоизация: Запазването на вече решени подзадачи помага за избягване на повторни изчисления, ускорявайки алгоритъма.

1. **Описание на задачи и решения с компилиран код**
   1. **Задача за раницата (Knapsack Problem)**

Задачата за раницата е комбинаторна оптимизационна задача, при която трябва да се определите предмети с различни тегла и стойности, които да бъдат поставени в раница с ограничен капацитет. Целта е максимизиране на общата стойност на предметите в раницата, съобразявайки се с обемните им ограничения.

Решение: Едно от ефективните решения на задачата за раницата включва използването на динамично програмиране. Създава се таблица, в която се съхранява максималната възможна стойност за всеки възможен обем на раницата и за всяка част от предметите. Запълването на тази таблица стъпка по стъпка, чрез съчетаване на оптимални решения за по-малки под проблеми, позволява намирането на общото оптимално решение за цялата задача.

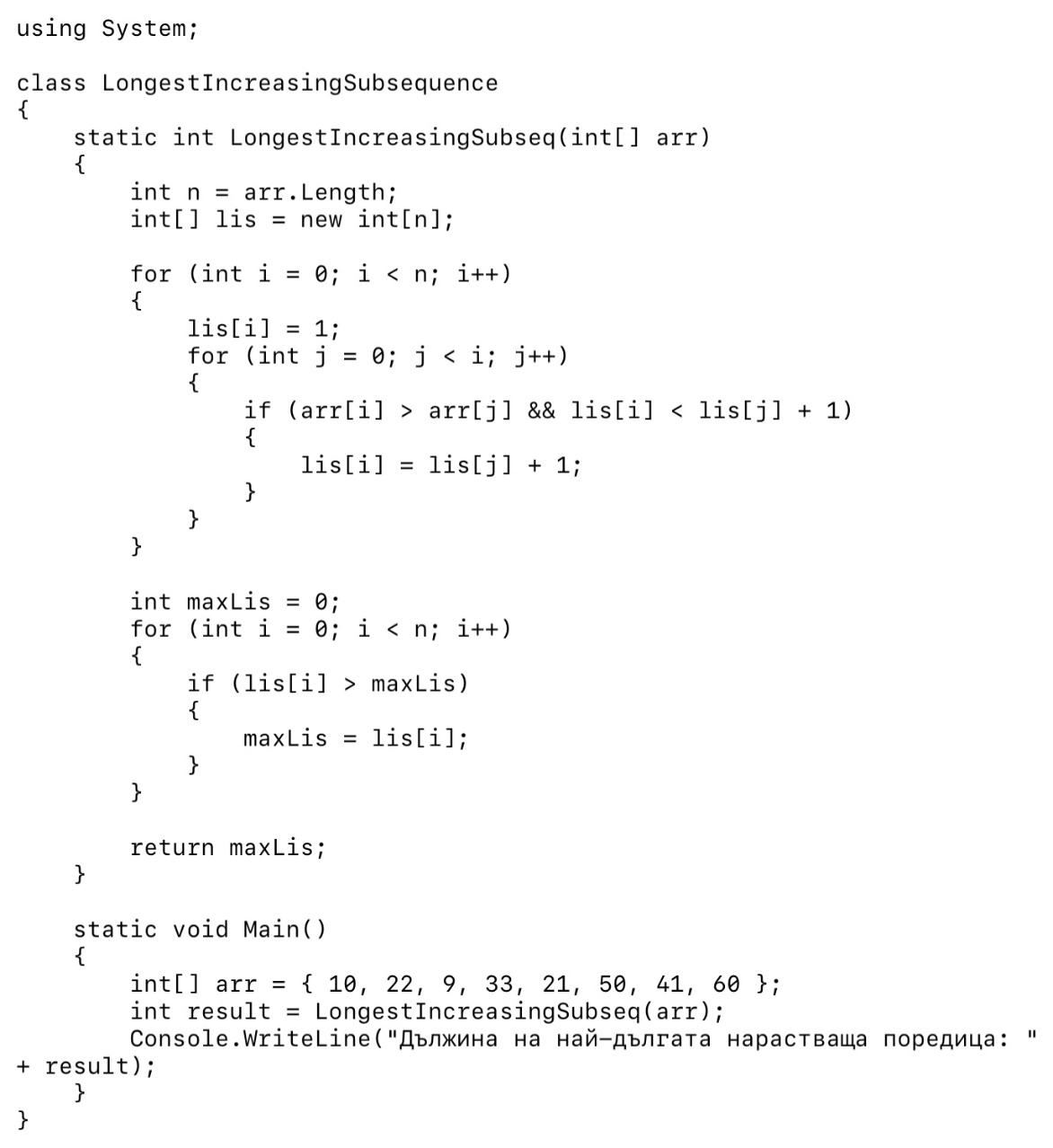
Примерен код:

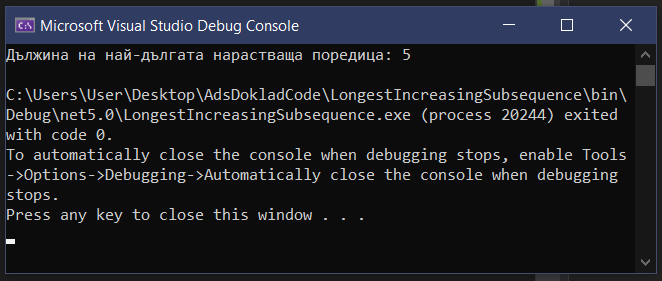
Компилиран код:

* 1. **Най-дълга нарастваща поредица (Longest Increasing Subsequence)**

Задачата за най-дълга нарастваща поредица (LIS) се формулира така: Даден е масив от числа, намерете дължината на най-дългата поредица, в която елементите са в нарастващ ред. Поредицата не е задължително да е последователна, но числата трябва да следват друг след друг в оригиналния масив.

Решение: Едно от ефективните решения на задачата за най-дълга нарастваща поредица използва динамично програмиране. Създава се масив, в който се съхранява информация за текущата дължина на LIS до всеки елемент от оригиналния масив.

****Примерен код:

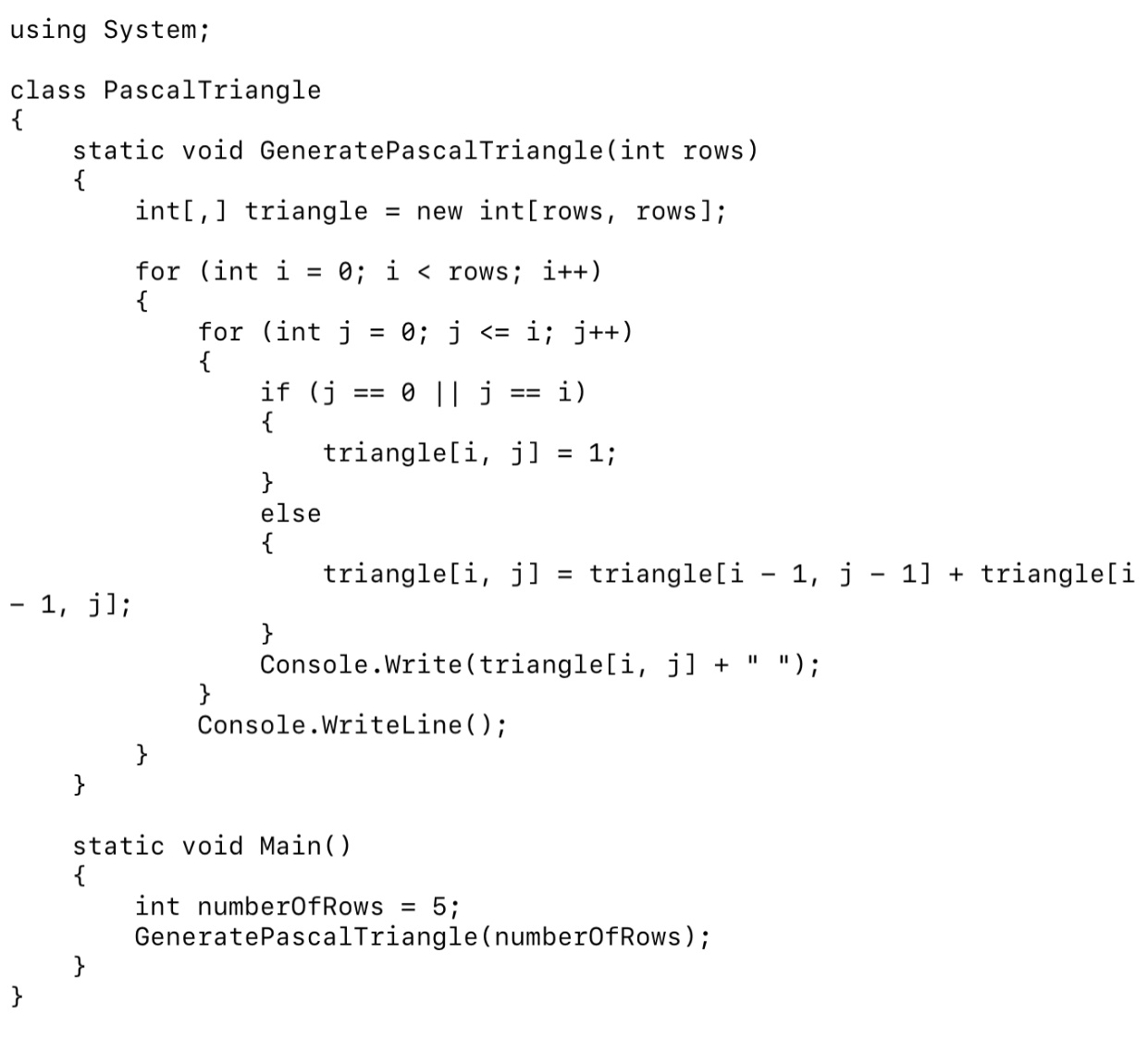
Компилиран код:

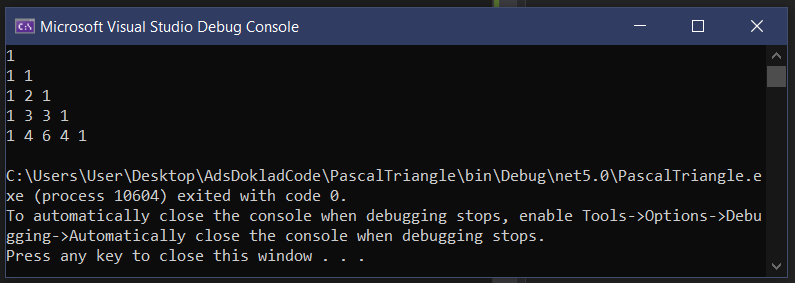
* 1. **Триъгълник на Паскал (Pascal’s Triangle)**

Триъгълникът на Паскал е математическа конструкция, която създава трънчета числа чрез комбинаторика. Във всяко число се получава като съберете двата числа, които са над него в предходния ред на триъгълника. Триъгълникът се представя като таблица от числа, като всяко число е сбор от двата числа над него в предходния ред.

Решение: Решението на задачата включва генериране и отпечатване на Триъгълника на Паскал. Създава се двумерен масив, който съхранява числата от триъгълника, като външният цикъл интегрира през всеки ред, а вътрешният цикъл изчислява стойностите за всяко число в текущия ред, използвайки правилата на Паскал.

Примерен код:



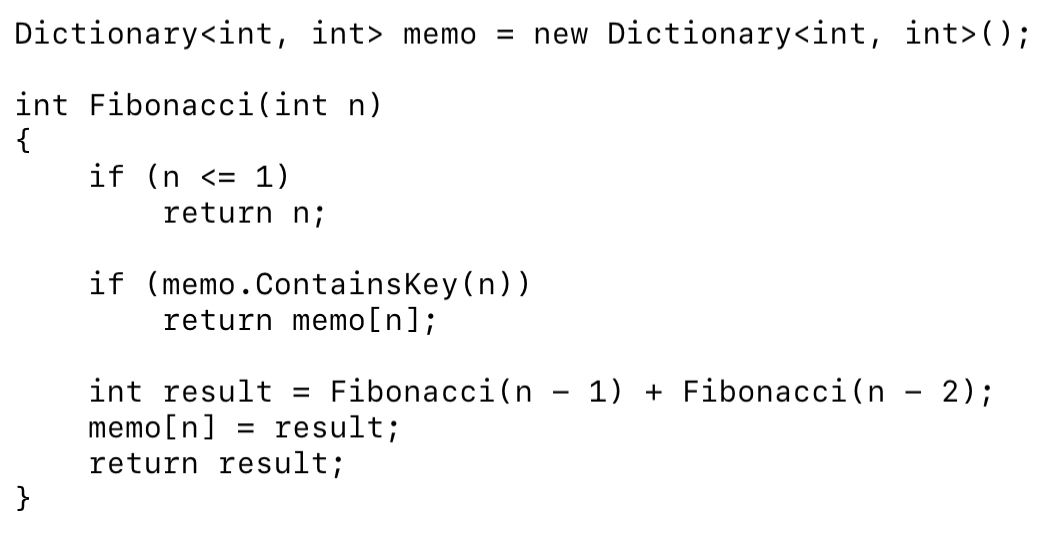
Компилиран код:

1. **Двумерни задачи за динамично програмиране. Примери**

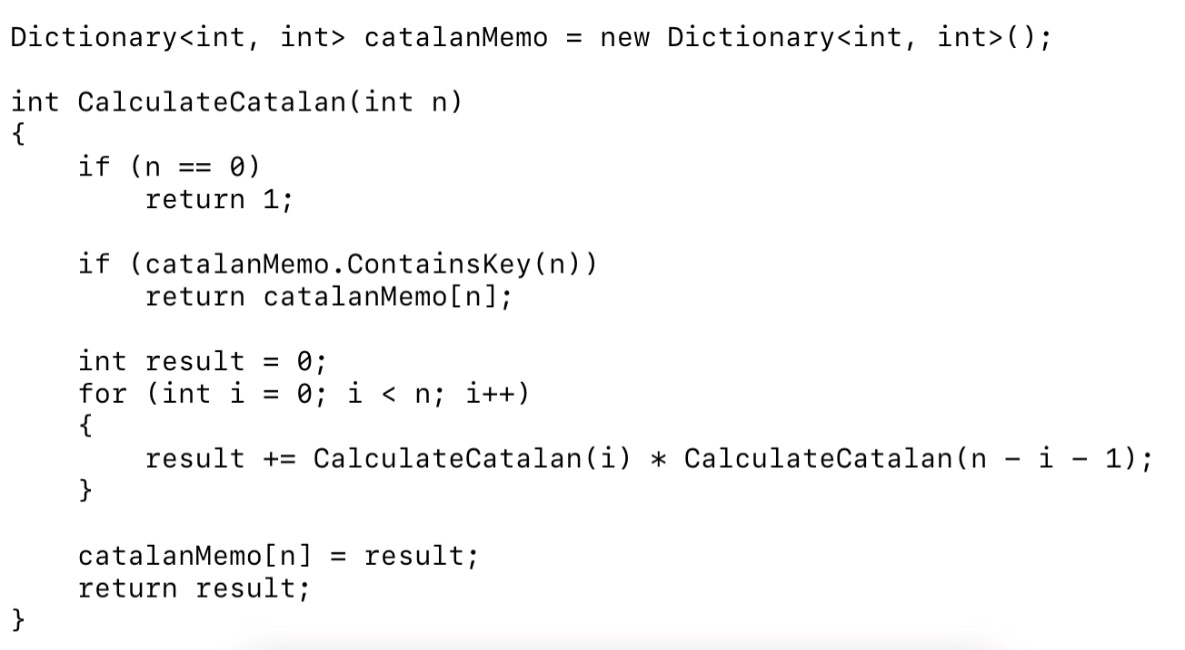
* **Матрично умножение (Matrix Chain Multiplication)**: Дадени са няколко матрици и целта е да се намери най-ефективният ред на тяхното умножение.
* Решение: Използвайки динамично програмиране, можем да оптимизираме броя на умноженията, като съхраняваме промежуточни резултати в двумерна таблица.
* **Най-дълъг общ под последователност (Longest Common Subsequence - LCS)**: Дадени са две последователности и целта е да се намери най-дългата обща под последователност.
* Решение: Използвайки двумерна таблица, се запълва матрица, която показва дължината на общите под последователности за всеки два префикса от входните последователности.
* **Най-дълга обща под дума (Longest Common Substring)**: Дадени са две думи и целта е да се намери най-дългата обща под дума (последователност от символи)
* Решение: Използвайки двумерна таблица, се запълва матрица, която показва дължината на общите поддуми за всеки два префикса от входните думи.
* **Минимално изрязване на вериги (Minimum Cut in a Chain of Matrices)**: Подобно на Matrix Chain Multiplication, но тук целта е намиране на минималния брой изрязвания за умножение на верига от матрици.
* Решение: Използвайки двумерна таблица, се определят минималните стойности за изрязвания за всеки интервал от матрици.
* **Най-дълъг път в триъгълник (Longest Path in Triangle)**: Даден е триъгълник от числа и целта е намери най-дългия път, като се движим само надолу и наляво.
* Решение: Използвайки двумерна таблица, се определят максималните суми за всеки възможен път, започващ от върха на триъгълника.

1. **Приложение на рекурсията в динамичното програмиране. Примери**

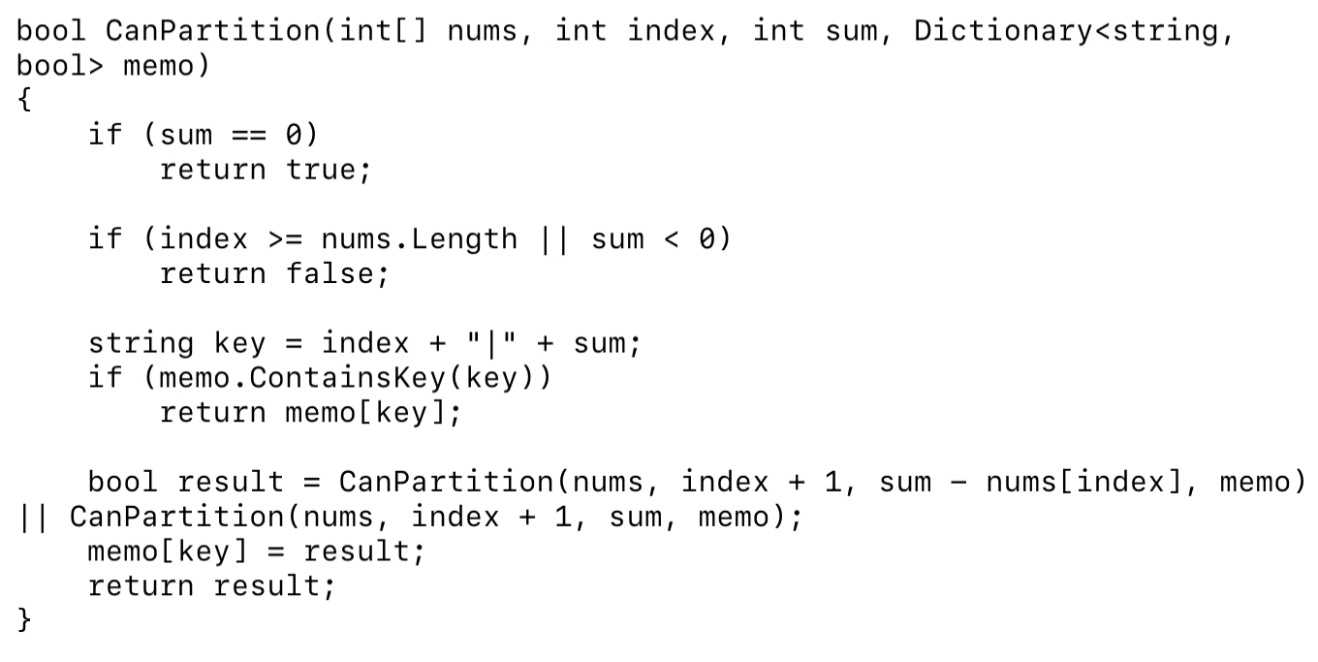
* **Число на Фибоначи**: Рекурсивно изчисление на числата на Фибоначи с мемоизация за избягване на повторни изчисления.



* **Число на Каталан**: Рекурсивно изчисление на числата на Каталан с мемоизация.



* **Разделяне на пари**: Рекурсивна проверка с мемоизация за разделянето на подмножества.



1. **Източници**

<https://en.wikipedia.org/wiki/Matrix_chain_multiplication#:~:text=Matrix%20chain%20multiplication%20(or%20the,of%20the%20matrix%20multiplications%20involved>.

<https://en.wikipedia.org/wiki/Longest_common_subsequence#:~:text=A%20longest%20common%20subsequence%20(LCS,positions%20within%20the%20original%20sequences>.

<https://en.wikipedia.org/wiki/Longest_common_substring>

<https://www.thealgorist.com/Algo/DynamicProgramming/MatrixChainMultiplication>

<https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_matrix_chain_multiplication.htm>

<https://www.geeksforgeeks.org/maximum-path-sum-triangle/>

<https://stackoverflow.com/questions/55778019/how-to-find-the-largest-path-sum-in-a-triangle-of-numbers>

<https://www.geeksforgeeks.org/program-for-nth-fibonacci-number/>

<https://www.programiz.com/c-programming/examples/fibonacci-series>

<https://www.geeksforgeeks.org/program-nth-catalan-number/>

<https://www.educative.io/answers/how-to-solve-the-partition-problem-with-dynamic-programming>

<https://en.wikipedia.org/wiki/Partition_problem>

<https://infoman.musala.com/files/view/static/anniversary/articles/Alexander.Georgiev_dynamic.programming.pdf/>

<https://www.spiceworks.com/tech/devops/articles/what-is-dynamic-programming/>

<https://www.geeksforgeeks.org/dynamic-programming/>

<https://en.wikipedia.org/wiki/Longest_increasing_subsequence>

<https://en.wikipedia.org/wiki/Knapsack_problem>

<https://en.wikipedia.org/wiki/Pascal%27s_triangle>