ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ
«Университет «Дубна»

ИНСТИТУТ СИСТЕМНОГО АНАЛИЗА И УПРАВЛЕНИЯ

Кафедра системного анализа и управления

КУРСОВАЯ РАБОТА

по дисциплине

«Теория принятия решений»

ТЕМА: принятие решения в случае модели динамического программирования

Выполнил: студент группы 2013

Ильин Михаил Алексеевич

(ФИО.)

(подпись студента)

Руководитель:

Ст. преп кафедры САУ Булякова И.А.

(ученая степень, ученое звание, занимаемая должность, ФИО)

Дата: 19.05.2022

Оценка: _____

(подпись руководителя)

Дубна, 2022

Оглавление

ВВЕДЕНИЕ

Для решения многих задач оптимизации, включающих большое число переменных и ограничений в виде неравенства, классический аппарат математики оказался непригодным. В результате для алгоритмов выгоднее стало разбивать задачу большой размерности на подзадачи, включающих всего по несколько переменных, и последующего решения общей задачи по частям. Для динамического программирования характерны специфические методы и приемы, применимые к операциям, в которых процесс принятия решения разбит на этапы (шаги). Именно эта идея стала основой при создании этого вида программирования.

Оптимизационные задачи встречаются почти во всех отраслях науки, техники, хозяйства и даже обычной жизни и других сферах человеческой деятельности. Именно поэтому круг применения динамического программирования очень широк.

Актуальность вышеупомянутого подхода программирования состоит в том, что в современной экономике широко используются оптимизационные методы, составляющие основу математического программирования.

ЦЕЛИ И ЗАДАЧИ

Цель курсовой работы: реализовать приложение «Подсчитывающая функция» в *Windows Forms*, которое рассчитывает количество вариантов складывания пластин в контейнеры на производстве.

Исходные данные: среда программирования *Visual Studio 2022*, знание языка программирования *C#*, необходимые теоретические знания реализации и применения динамического программирования.

Модельное представление: программа, которая должна помогать пользователю в расчете вариантов укладки пластин в контейнеры. С помощью написанных алгоритмов и методов пользователь может задавать параметры вместимости контейнеров и условия укладки пластин, количество которых задается так же пользователем индекс контейнера для одиночного отображения. После этого пользователь увидит вычисленные обработанные данные на форме.

Средства реализации: приложение *Windows Forms (.NET Framework)* в *Visual Studio 2022*, язык программирования *C#*.

Ожидаемый результат: приложение, которое корректно высчитывает и отображает выходные данные.

Критерий оценки результата: корректное число вариантов для всех вариаций компоновки элементов.

Задачи, которые необходимо решить в данной работе:

- ✓ краткое отображение условия задачи,
- ✓ ввод данных для работы с алгоритмом,
- ✓ реализация алгоритма подсчета вариаций,
- ✓ проверка работы алгоритма при различных входных параметрах.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

ОСНОВНАЯ ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Динамическое программирование является одним из разделов оптимального программирования. Методами динамического программирования решаются вариантные оптимизационные задачи с заданными критериями оптимальности, с определенными связями между переменными и целевой функцией, выраженными системой уравнений или неравенств.

В основе метода данного программирования лежит принцип оптимальности, впервые сформулированный в 1953 г. американским математиком Р. Э. Беллманом: каково бы ни было состояние системы в результате какого-либо числа шагов, на ближайшем шаге нужно выбирать управление так, чтобы оно в совокупности с оптимальным управлением на всех последующих шагах приводило к оптимальному выигрышу на всех оставшихся шагах, включая выигрыш на данном шаге. При решении задачи на каждом шаге выбирается управление, которое должно привести к оптимальному выигрышу. Если считать все шаги независимыми, тогда оптимальным решением будет то, которое обеспечит максимальный выигрыш именно на данном шаге.

Динамическое программирование можно использовать как для решения задач, связанных с динамикой процесса или системы, так и для статических задач, связанных, например, с распределением ресурсов. Это значительно расширяет область применения динамического программирования для решения задач управления. А возможность упрощения процесса решения, которая достигается за счет ограничения области и количества, исследуемых при переходе к очередному этапу вариантов, увеличивает достоинства этого комплекса методов.

Вместе с тем динамическому программированию свойственны и недостатки. Прежде всего, в нем нет единого универсального метода решения. Практически каждая задача, которая решается данным методом, характеризуется своими особенностями и требует проведения поиска наиболее приемлемых методов для ее решения. Кроме того, большие объемы и трудоемкость решения многошаговых задач, имеющих множество состояний и переменных, приводят к необходимости отбора задач малой размерности либо использования сжатой информации.

Однако, несмотря на все достоинства, динамическому программированию свойственны и недостатки. Прежде всего, в нем нет единого универсального метода решения. Практически каждая задача, решаемая данным методом, характеризуется своими особенностями и требует проведения анализа переменных для корректного решения задачи. Кроме того, большие

объемы и трудоемкость решения многошаговых задач, имеющих множество состояний, приводят к необходимости отбора задач малой размерности либо использования неполной или сжатой информации. Последнее достигается с помощью методов анализа вариантов и переработки исходных состояний. Для процессов с непрерывным временем рассматривается как предельный вариант дискретной схемы решения. Получаемые при этом результаты практически совпадают с теми, которые получаются методами максимума Беллмана.

Динамическое программирование применяется для решения задач, в которых поиск оптимума возможен при поэтапном подходе, например, распределение капитальных вложений между возможными методами их использования; разработка плана управления запасами и размерами предполагаемых заказов; составления расчета для текущего и капитального ремонтов оборудования и/или его замены; поиск кратчайших расстояний на транспортной сети; составления последовательности результатов операции и вычислений и т.д.

Динамическое программирование в теории управления и теории вычислительных систем — способ решения сложных задач путём разбиения их на более простые подзадачи. Он применим к задачам с оптимальной подструктурой, выглядящим как набор перекрывающихся подзадач, сложность которых чуть меньше исходной. При этом отличительной особенностью является их решение по этапам, через фиксированные интервалы, промежутки времени, что и определило появление термина динамическое программирование. Планирование каждого шага должно проводиться с учетом общей выгоды, получаемой по завершении всего процесса, что и позволяет оптимизировать конечный результат по выбранному критерию.

Таким образом, динамическое программирование в широком смысле представляет собой оптимальное управление процессом, посредством изменения управляемых параметров на каждом шаге, и, следовательно, воздействуя на ход процесса, изменяя на каждом шаге состояние системы.

МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

Формализация задачи: при переработке радиоактивных материалов на химическом заводе образуются отходы двух видов, которые прессуются в пластины. Первый вид – особо опасные (относятся к типу 1), второй вид – безопасные (относятся ко 2 типу). Для их хранения используются одинаковые контейнеры. После помещения отходов в них, пластины складываются в стопки. Стопка считается опасной для окружающей среды, если в ней идёт подряд более вводимого пользователем количества пластин типа 1. Стопка считается безопасной, если она не является опасной для природы. Необходимо для заданного количества контейнеров N, заданного количества пластин в контейнере m и ограничении количества подряд идущих платин типа 1 n определить количество возможных типов безопасных стопок.

Обозначим входные данные:

N – общее количество контейнеров для отходов (вводится пользователем в *textBox*, варьируется в указанном на форме диапазоне),

n – количество пластин в контейнере (вводится пользователем в *textBox,* в указанном на форме диапазоне),

m – количество пластин, которое может идти подряд в контейнере (так же вводится пользователем и варьируется).

Рассмотрим частный случай задачи для небольших переменных и выявим зависимость (рис. 1). $N = 3$, $n = 3$, $m = 2$. Далее такая конфигурация будет считаться начальным положением и обозначаться НП.

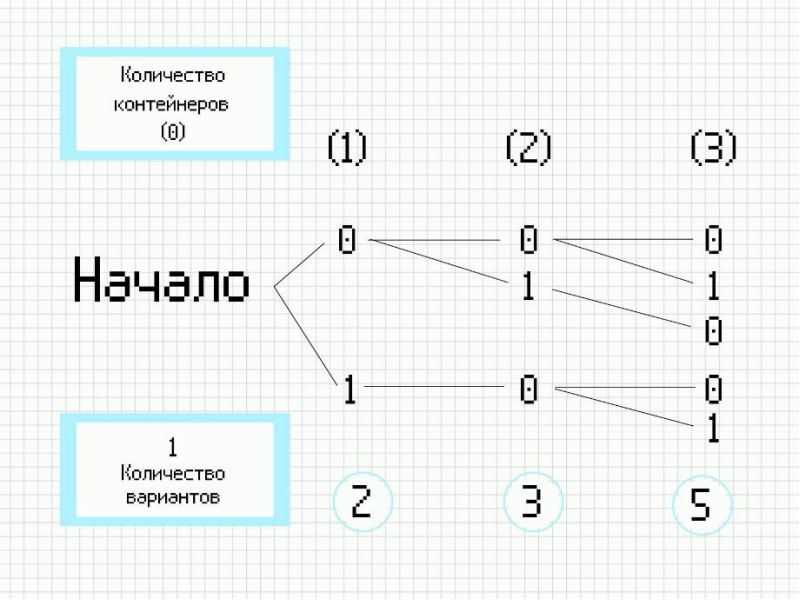


Рис. 1. Решение задачи распределения пластин для случая *N*(3), *n*(от 1 до 3) и *m*(2).

Чтобы подсчитать количество вариантов при 0 контейнеров, не нужны входные данные, так как при 0 контейнеров есть только 1 вариант – ни пластин, ни контейнеров.

Далее для 1 контейнера есть выбор, положить ли туда пластину типа 1 или 2, значит вариантов уже 2. То есть, до $m - 1$ контейнеров сохраняется пропорция:

$$result_i = result_{i-1} * 2,$$

где *result* – количество вариантов для контейнера $i - 1$.

Потом, для 2 контейнера уже ставиться ограничение, так как нельзя допустить попадания m пластин подряд, поэтому нам необходимо взять другую формулу для вычисления:

$$result_i = result_{i-1} + result_{i-2},$$

формула для контейнеров индекса *m* и далее.

Соответственно, для расчета количества вариантов, необходимо знать все предыдущие m шагов назад данные.

Весь алгоритм мы проводим $N * n - m$ раз, для того чтобы получить последовательность чисел, зависящих друг от друга.

Для функции расчета по индексу проводиться те же вычисления, но индекс контейнера записывается в переменную, а значения выписываются для заданного контейнера сразу.

Таким образом, у нас создается образ стопок и контейнеров со значениями, который для наглядности отрисовывается на главной форме и форме представления краткой модели.

ПРАКТИЧЕСКАЯ ЧАСТЬ

ИСХОДНАЯ ФОРМА

Запустив приложение, мы видим открывшуюся форму (рис. 2), на которой есть кнопки для взаимодействия и вызова функций, несколько *labels* с указанием ограничений для исходных данных, а также панель для ввода индекса, по которому нам необходимо вычислить функцию.

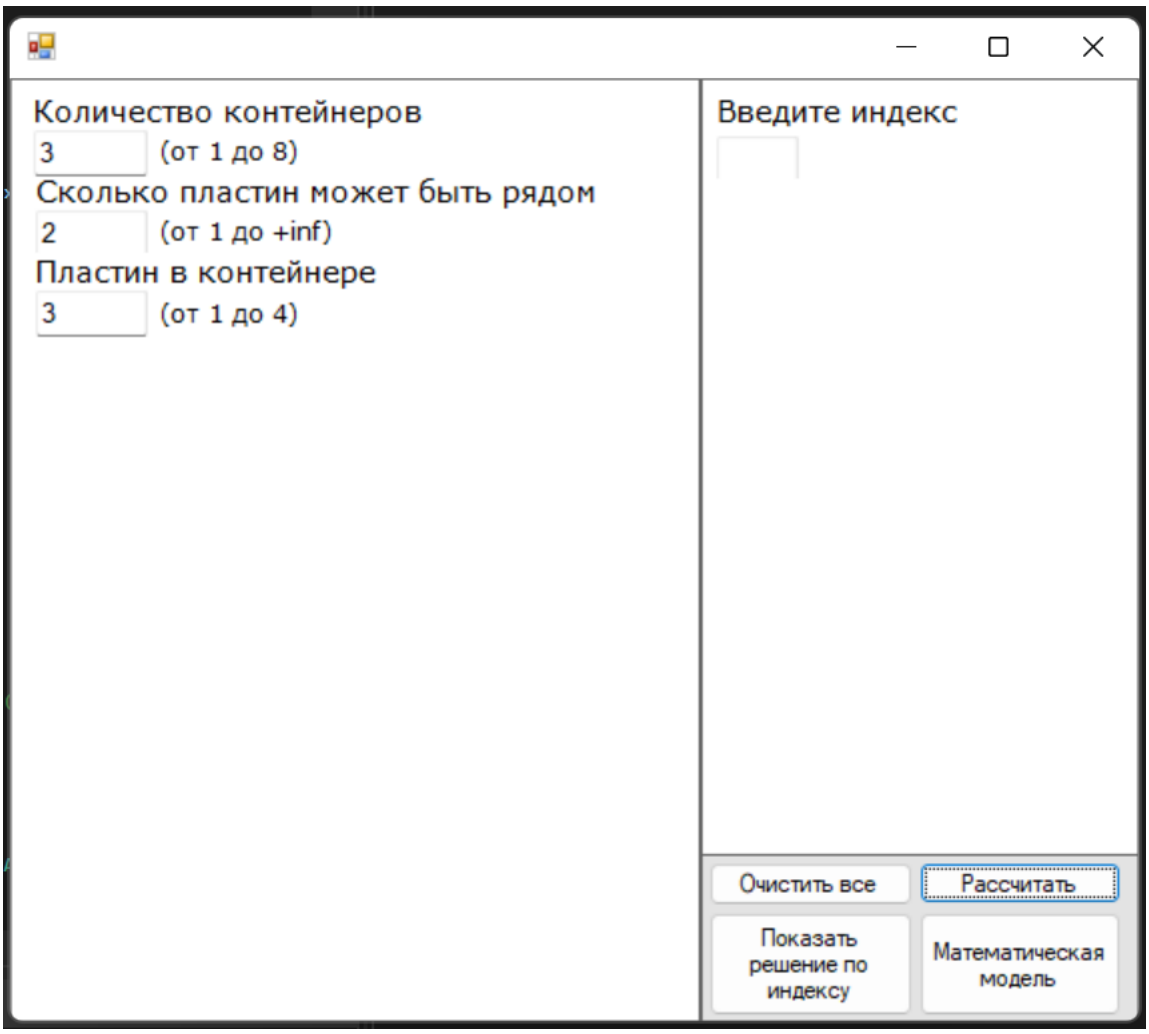


Рис. 2. Стартовое окно программы.

Кнопка «Математическая модель» открывает новую форму (рис. 3), на которой представлена краткая информация о задаче, пример расположения пластин в контейнере для удобного представления задачи.

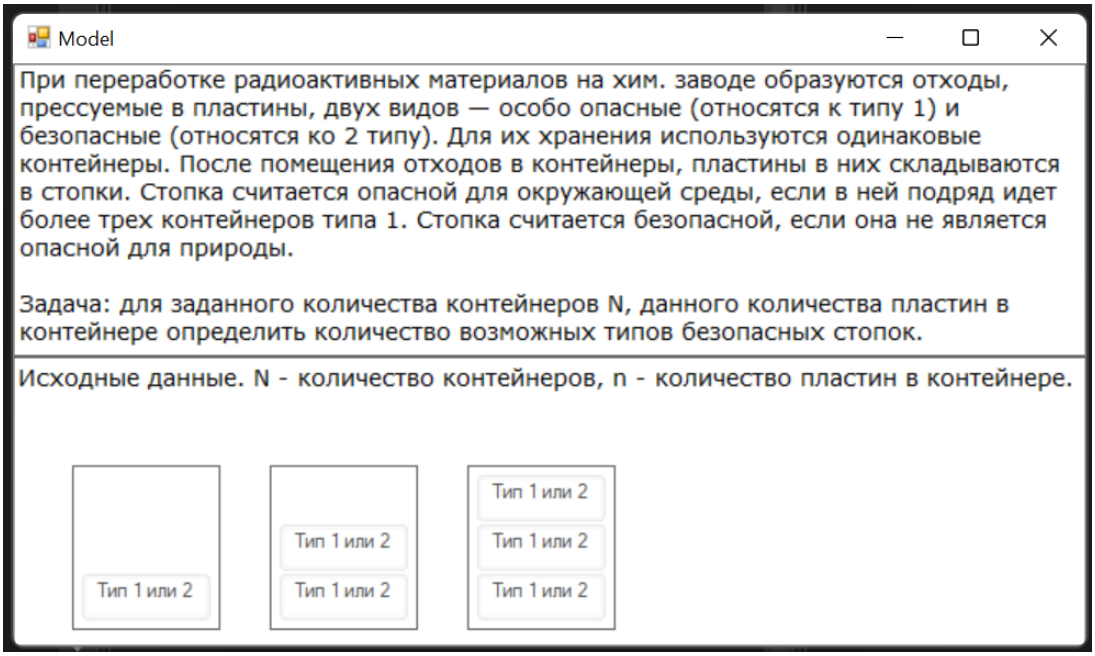


Рис. 3. Вторая форма приложения.

Кнопка «Рассчитать» отвечает за обработку, вычисление, запись, вывод на форму данных, при вышевводимых исходных.

Кнопка «Показать решение по индексу» отображает конкретное решение для вводимого индекса контейнера без отображения других.

А кнопка «Очистить все» будет вызывать функцию очистки полей, возвращающую программу к НП и обнуляющую значения и отображение созданных элементов.

Для вычисления количества возможных вариантов будет создаваться массив, который будет хранить все исходные 4 значения (*isit*), а также массив, который хранит полученные значения (*res*). Для массива *isit* зададим НП, а значение поля индекс остается пустым до того момента, пока пользователь сам не введет значение.

РЕАЛИЗАЦИЯ АЛГОРИТМА ВЫЧИСЛЕНИЯ

Как было выше упомянуто, нам необходимо заполнить массив *res* до индекса m элементами. Первый всегда равен константе 1. Остальные заполняются вышеупомянутым способом (рис. 4).

```
int product = isit[0] * isit[2];

int[] res = new int[product + 1]; // создаем массив для N * n + 1 элементов
int sum = 0;
res[0] = 1;

for (int i = 1; i < isit[1]; i++)
{
    res[i] = res[i - 1] * 2; // массив контейнера - веса пластин изначально 0
}
```

Рис. 4. Массив *res. Н*ачальные элементы.

Далее нам необходимо создать алгоритм, который будет подсчитывать число вариантов на определенном шаге и записывать в массив *res.* Его реализация представлена на рис.5.

```
for (int i = isit[1]; i < product + 1; i++) // все контейнеры
{
    for (int lb = i - isit[1]; lb < i; lb++)
    {
        sum += res[lb];
    }
    res[i] = sum;
    sum = 0;
}

for (int n = 0; n < isit[0]; n++)
{
    label.Text += (n + 1).ToString() + ") ";
    for (int m = 1; m < isit[2] + 1; m++)
    {
        label.Text += res[m + isit[2] * n].ToString() + ' ';
    }
    label.Text += "\n";
    label.Text += "\n";
}
```

Рис. 5. Запись в массиве *res.*

Этот алгоритм не только вычисляет и записывает значение в массив *res*, но и выводит результат на экран (рис. 6).

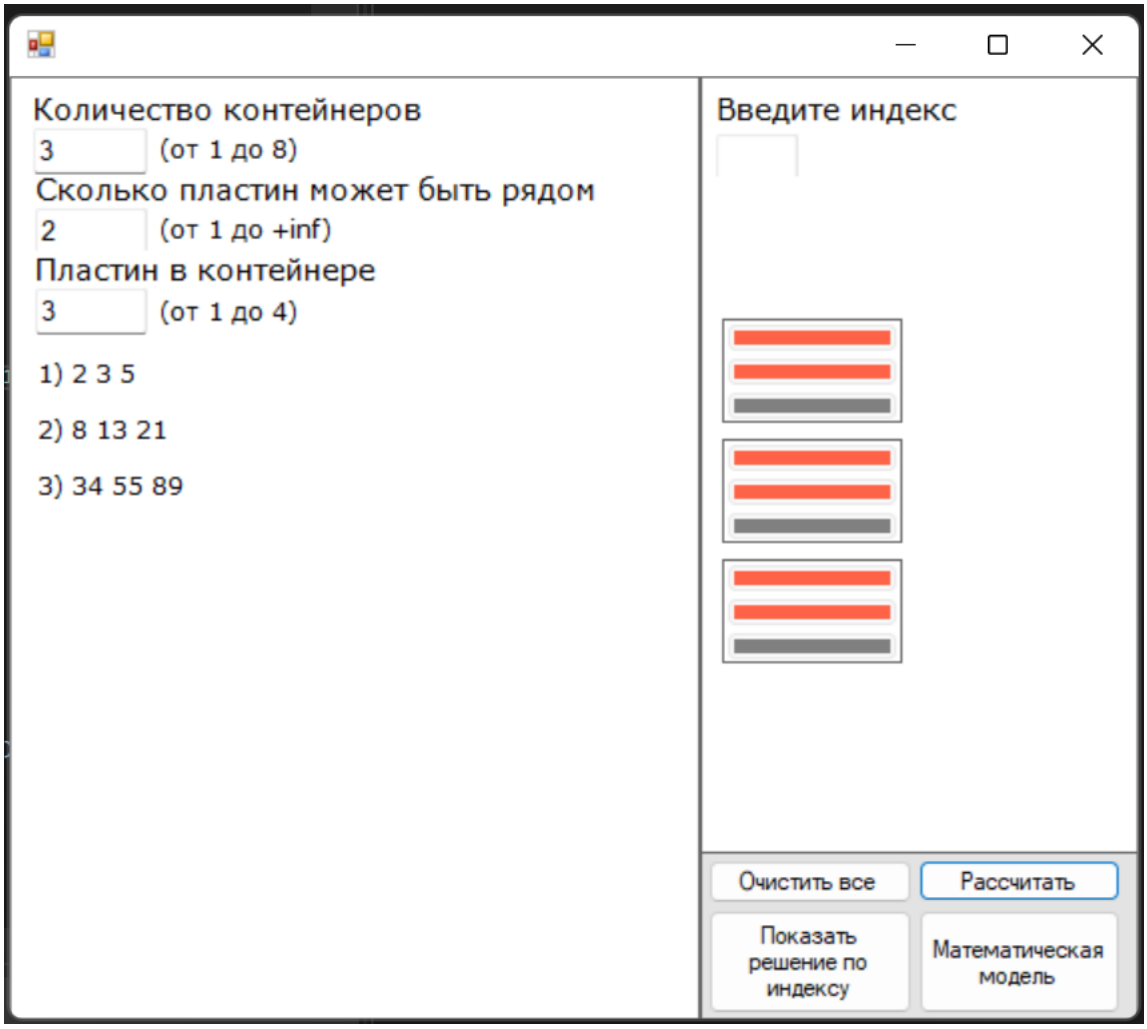


Рис. 6. Результат работы программы.

Также, на этом рисунке видны изображенные пластины, красно-оранжевым отображено наибольшее количество идущих рядом пластин типа 1, серым цветом – пластины типа 2.

Для реализации вывода по индексу, используется почти тот же самый алгоритм, только запоминающий номер контейнера. При выводе, мы отображаем только элементы полученного нами индекса контейнера (рис. 7).

```
for (int i = isit[1]; i < product + 1; i++) // все контейнеры
{
    for (int lb = i - isit[1]; lb < i; lb++)
    {
        sum += res[lb];
    }
    res[i] = sum;
    sum = 0;

}
label11.Text += index.ToString() + ") ";
for (int m = index * isit[2] - isit[2] + 1; m < index * isit[2] + 1; m++)
{
        label11.Text += res[m].ToString() + ' ';
}
```

Рис. 7. Алгоритм по индексу.

И получаем выходной результат выполнения:

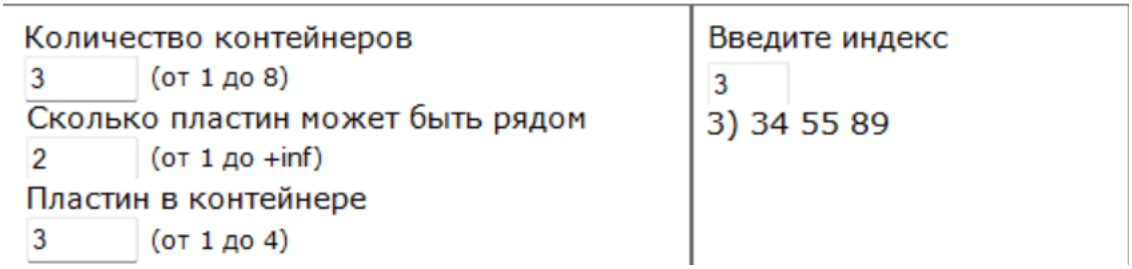
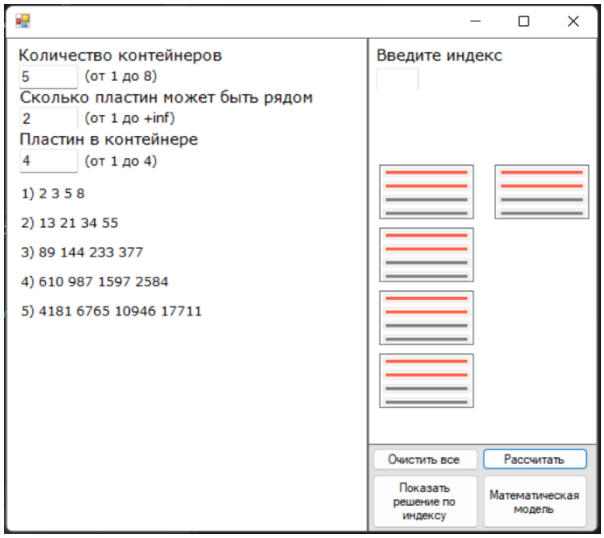
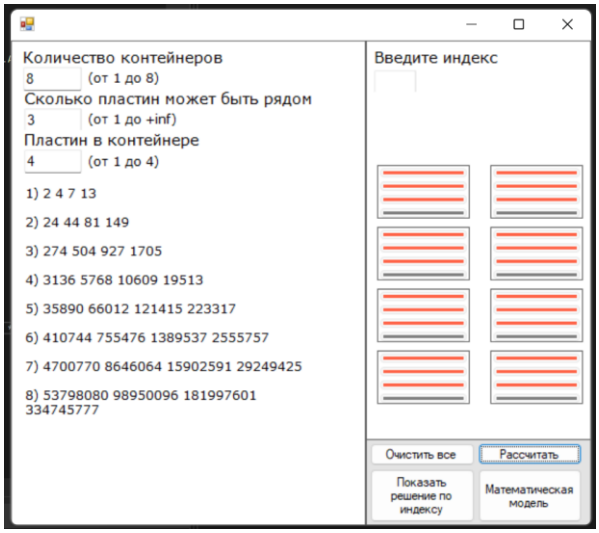


Рис. 8. Пример работы алгоритма по индексу.

Теперь проверим работоспособность написанного приложения для различных данных и убедимся в корректности полученных ответов:





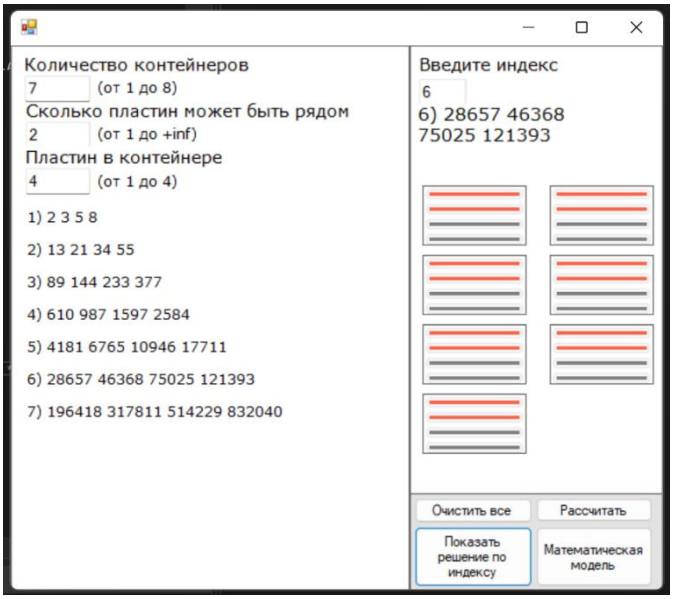


Рис. 9, 10, 11. Результат работы программы про различных входных данных.

ВЫВОД

В данной курсовой работе был рассмотрен метод динамического программирования, представлено собственное приложение. В нем был реализован алгоритм для подсчета вариантов сложения пластин. Работа алгоритма вычисления была продемонстрирована на различных пользовательских данных. Было выявлено, что его работа, выходные данные и отображение – корректны, не вызывают ошибок у пользователя, реализация к пониманию.

СПИСОК ЛИТЕРАТУРЫ

1. Лекции «Теория принятия решений» Черемисина Е.Н. 2022.
2. Динамическое программирование. Классические задачи [Электронный ресурс]. 2022. URL: https://habr.com/ru/post/113108/ (дата обращения 12.05.2022).
3. Уравнение Беллмана [Электронный ресурс]. 2022. URL: https://ru.wikipedia.org/wiki/%D0%A3%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D0%91%D0%B5%D0%BB%D0%BB%D0%BC%D0%B0%D0%BD%D0%B0 (дата обращения 12.05.2022).