|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 8** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Алгоритмические стратегии или методы разработки алгоритмов. Перебор и методы его сокращения»** | |
|  | |
| Выполнил студент группы ИКБО-06-21 | Школьник Т.О. |
| Принял преподаватель | Филатов А.С. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2022

# **Цель работы**

Получить навыки применения методов, позволяющих сократить число переборов в задачах, которые могут быть решены только методом перебора всех возможных вариантов решения.

# **Постановка задачи**

1. Разработать алгоритм решения задачи с применением метода, указанного в варианте и реализовать программу.
2. Оценить количество переборов при решении задачи стратегией «в лоб» - грубой силы.
3. Привести анализ снижения числа переборов при применении метода.
4. Составить отчет, отобразив в нем описание выполнения всех этапов разработки, тестирования и код всей программы со скриншотами результатов тестирования.

Вариант №9. Условие задания:

|  |  |  |
| --- | --- | --- |
| №  Варианта | Задача | Метод |
| 9 | Треугольник имеет вид, представленный на рисунке. Напишите программу, которая вычисляет наибольшую сумму чисел, расположенных на пути, начинающемся в верхней точке треугольника и заканчивающегося на основании треугольника. | Динамическое программирование |

# **Решение**

Динамическое программирование — способ решения сложных задач путём разбиения их на более простые подзадачи. Динамическое программирование применимо не ко всем задачам, а лишь к тем, которые обладают «оптимальной структурой». Оптимальная структура означает, что задача может быть разбита на несколько аналогичных задач меньшего размера, при этом для решения конечной задачи могут быть использованы результаты решения меньших задач.

Оценим асимптотику решения данной задачи методом грубой силы. Давайте переберем все возможные пути от вершины треугольника к его основанию и из получившихся сумм выберем максимальную. На каждом шаге можно перейти к i-му или к i+1-му элементу в следующей строке. Всего шагов будет , где – высота треугольника. Итоговая временная сложность такого подхода –

Оптимизируем наивное решение. Обойдем треугольник снизу вверх. Базой динамики будет являться последняя строчка, поскольку для этих элементов ответ равен им самим. Тогда для элемента будем выбирать максимальное значение из элементов , где будет храниться наибольшая сумма, которую можно набрать по пути от этих элементов до нижней строчки.

Временная сложность такого подхода – , пространственная – или, если модификация входных данных не допускается, то для хранения массива максимальных сумм предыдущей строчки.

Описание работы программы:

Пользователь вводит высоту треугольника и матрицу, задающую треугольник (Рисунок 1).

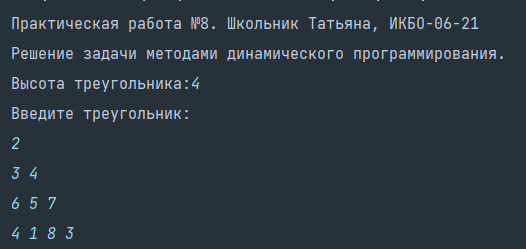


Рисунок 1. Пример пользовательского ввода

После ввода данных вызывается функция solution (Рисунки 2, 3), которая реализует указанный выше алгоритм и возвращает число – ответ на вопрос задачи.

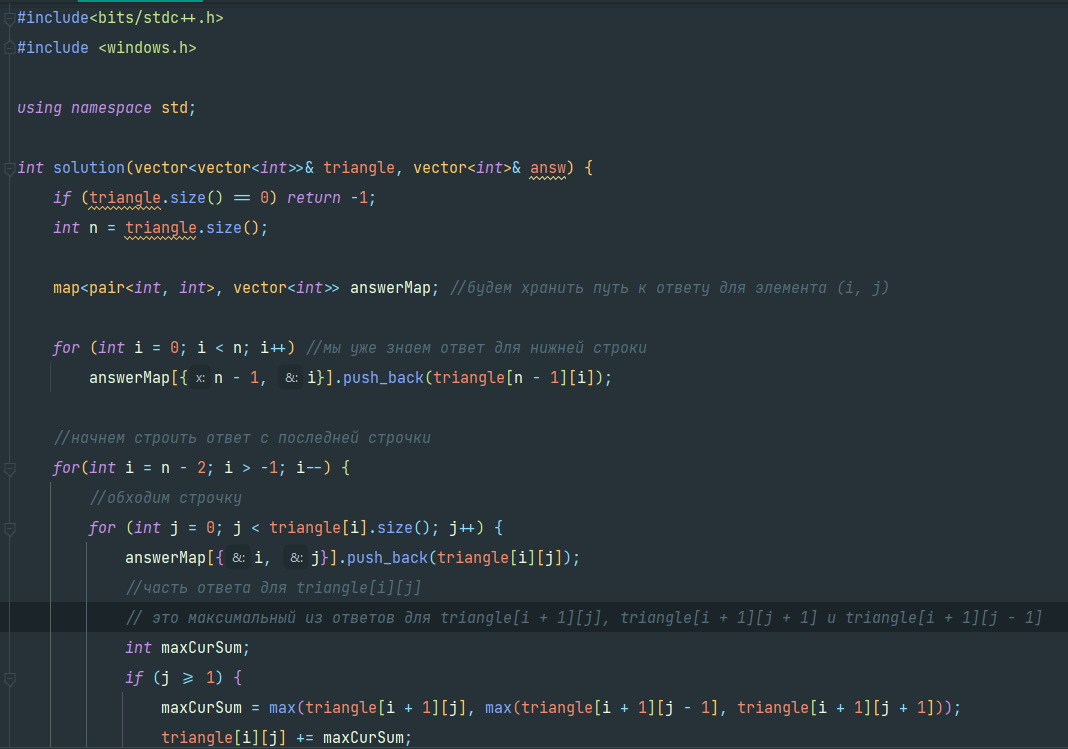


Рисунок 2. Фрагмент кода функции solution

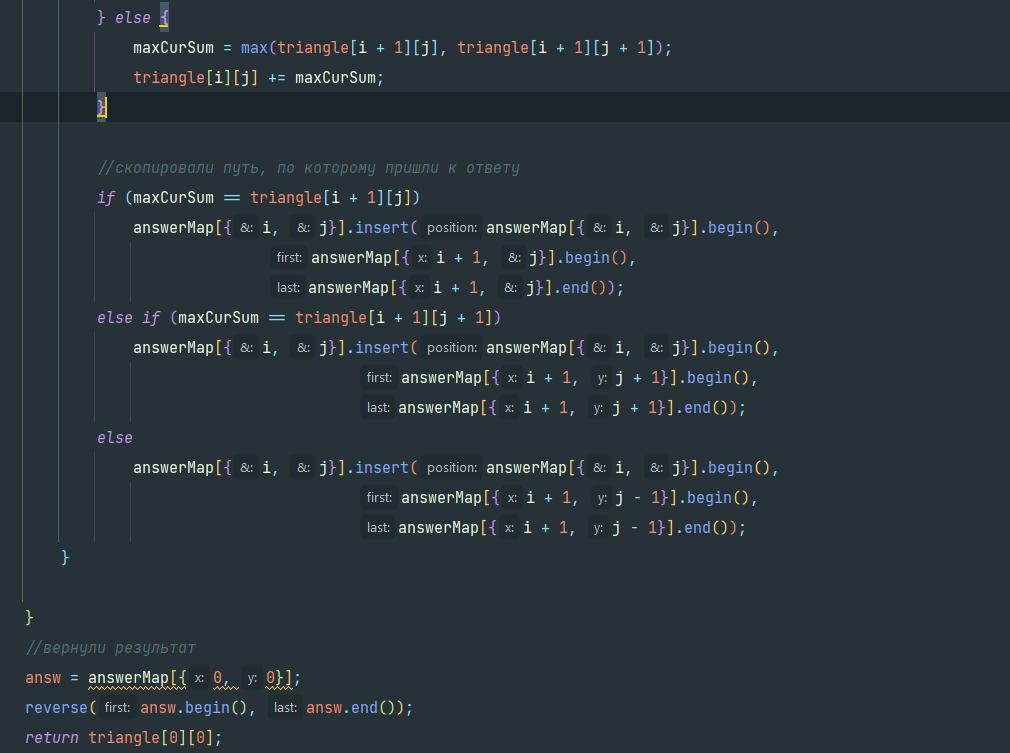


Рисунок 3 – Фрагмент кода функции solution

В консоль выводится это значение и результирующий путь от вершины треугольника до его основания (Рисунок 4).

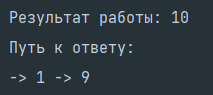


Рисунок 4. Пример вывода программы

# **Тестирование**

Проверка работоспособности программы (Рисунки 5 – 9).

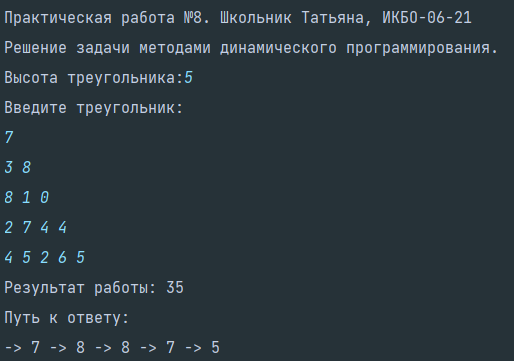


Рисунок 5. Тестирование программы

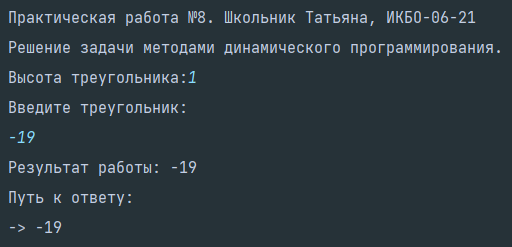


Рисунок 6. Тестирование программы

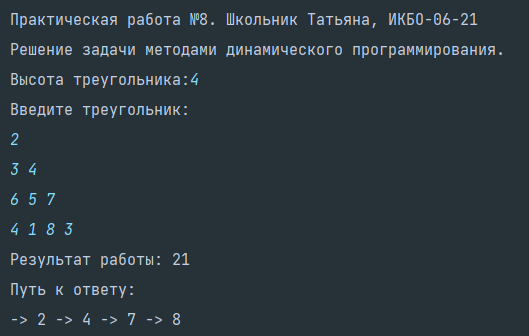


Рисунок 7. Тестирование программы

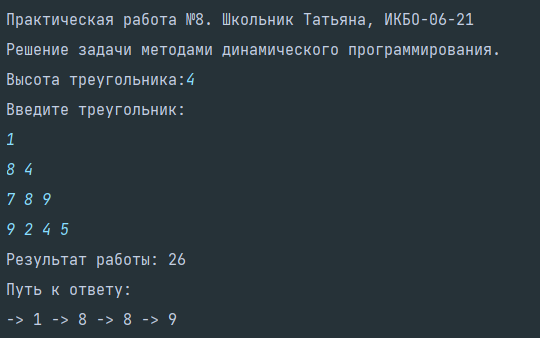


Рисунок 8. Тестирование программы

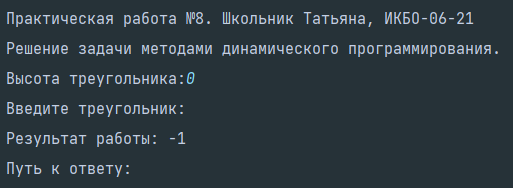


Рисунок 9. Тестирование программы

Из результатов выполнения программы видно корректную отработку программы согласно условиям задачи.

# **Вывод**

В результате выполнения работы были получены навыки сокращения времени работы программы посредством применения методов динамического программирования для тех задач, которые могут быть решены изначально только при помощи метода полного перебора.

1. **Исходный код программы**

main.cpp

|  |
| --- |
| #include<bits/stdc++.h>  #include <windows.h>  using namespace std;  int solution(vector<vector<int>>& triangle, vector<int>& answ) {  if (triangle.size() == 0) return -1;  int n = triangle.size();  map<pair<int, int>, vector<int>> answerMap; //будем хранить путь к ответу для элемента (i, j)  for (int i = 0; i < n; i++) //мы уже знаем ответ для нижней строки  answerMap[{n - 1, i}].push\_back(triangle[n - 1][i]);  //начнем строить ответ с последней строчки  for(int i = n - 2; i > -1; i--) {  //обходим строчку  for (int j = 0; j < triangle[i].size(); j++) {  answerMap[{i, j}].push\_back(triangle[i][j]);  //часть ответа для triangle[i][j]  // это макcимальный из ответов для triangle[i + 1][j] и triangle[i + 1][j + 1]  int maxCurSum;  if (j >= 1) {  maxCurSum = max(triangle[i + 1][j], max(triangle[i + 1][j - 1], triangle[i + 1][j + 1]));  triangle[i][j] += maxCurSum;  } else {  maxCurSum = max(triangle[i + 1][j], triangle[i + 1][j + 1]);  triangle[i][j] += maxCurSum;  }  //скопировали путь, по которому пришли к ответу  if (maxCurSum == triangle[i + 1][j])  answerMap[{i, j}].insert(answerMap[{i, j}].begin(),  answerMap[{i + 1, j}].begin(),  answerMap[{i + 1, j}].end());  else if (maxCurSum == triangle[i + 1][j + 1])  answerMap[{i, j}].insert(answerMap[{i, j}].begin(),  answerMap[{i + 1, j + 1}].begin(),  answerMap[{i + 1, j + 1}].end());  else  answerMap[{i, j}].insert(answerMap[{i, j}].begin(),  answerMap[{i + 1, j - 1}].begin(),  answerMap[{i + 1, j - 1}].end());  }  }  //вернули результат  answ = answerMap[{0, 0}];  reverse(answ.begin(), answ.end());  return triangle[0][0];  }  int main() {  SetConsoleOutputCP(CP\_UTF8);  vector<vector<int>> triangle;  vector<int> answ;  int n;  cout << "Практическая работа №8. Школьник Татьяна, ИКБО-06-21" << "\n";  cout << "Решение задачи методами динамического программирования." << "\n";  cout << "Высота треугольника:";  cin >> n;  cout << "Введите треугольник: " << "\n";  for (int i = 1; i <= n; i++) {  vector<int>row(i);  for (int j = 0; j < i; j++) cin >> row[j];  triangle.push\_back(row);  }  cout << "Результат работы: " << solution(triangle, answ) << "\n";  cout << "Путь к ответу:" << "\n";  for (auto value : answ) {  cout << "-> " << value << " ";  }  cout << "\n";  return 0;  } |