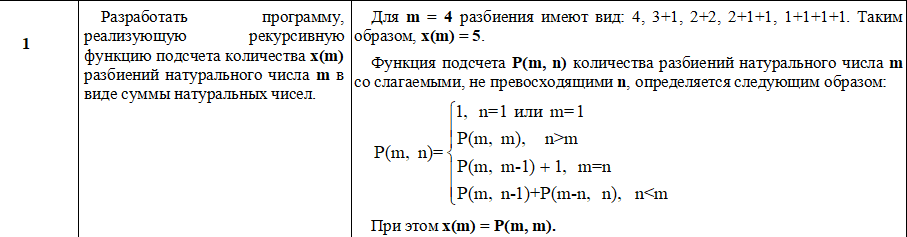
Банкузов Михаил Лабораторная работа №10 Вариант 1



#include <iostream>

using namespace std;

// Рекурсивная функция для подсчета количества разбиений

int countPartitions(int n, int m) {

// Базовый случай

if (n == 0) return 1;

// Рекурсивный случай

int count = 0;

for (int i = 1; i <= m && i <= n; i++) {

count += countPartitions(n - i, i);

}

return count;

}

int main() {

setlocale(LC\_ALL, "Russian");

int m;

cout << "Введите натуральное число m: ";

cin >> m;

// Вызываем функцию для подсчета количества разбиений

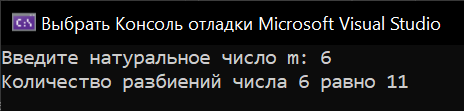
int partitions = countPartitions(m, m);

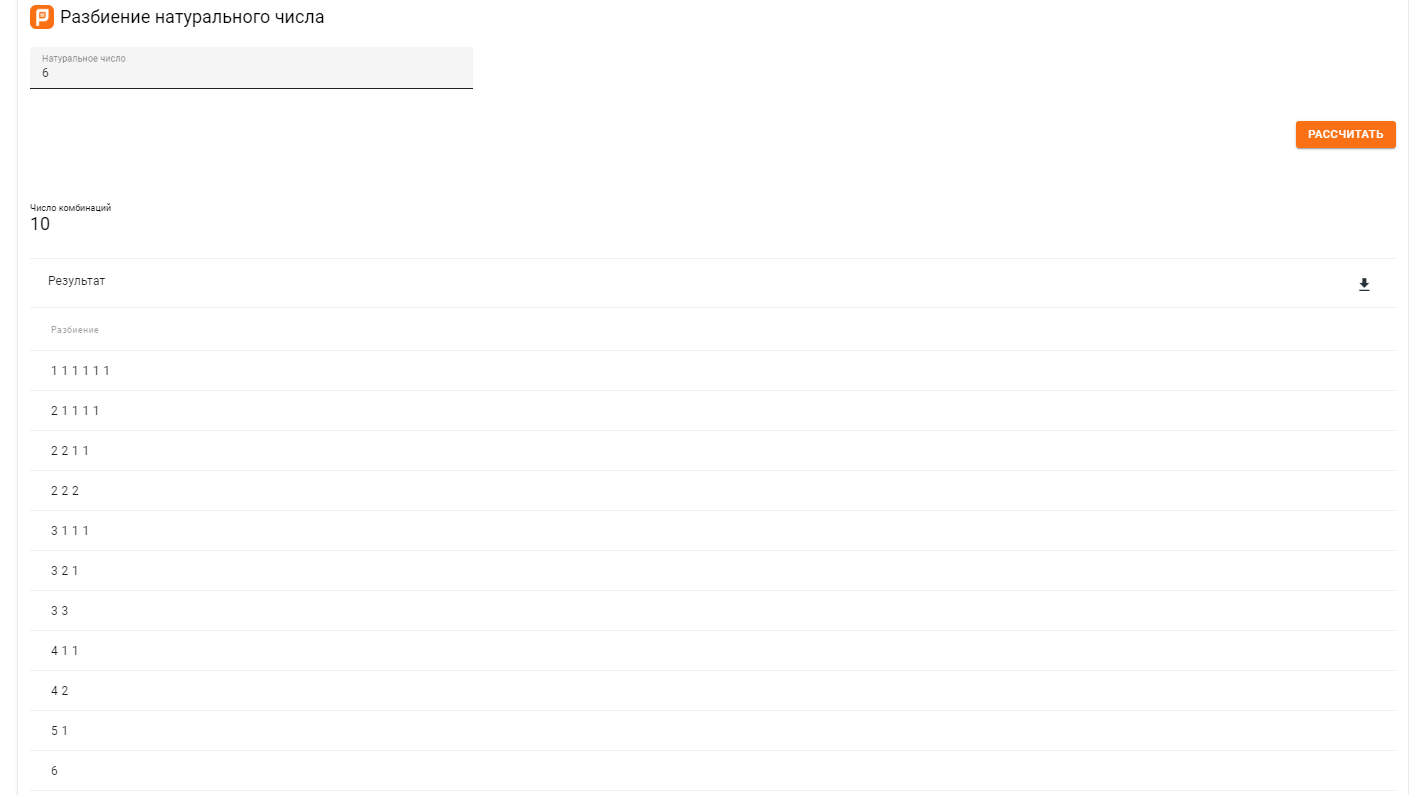
// Выводим результат

cout << "Количество разбиений числа " << m << " равно " << partitions << endl;

return 0;

}





Задание 2





#include <iostream>

#include <cmath>

using namespace std;

// Функция nestedSqrt используется для рекурсивного вычисления значения формулы вложенных квадратных корней

double nestedSqrt(int n, int k) {

// Если достигнут конечный уровень вложенности (k == n), возвращаем значение корня из соответствующей суммы

if (k == n) {

return sqrt(1 + (n + n));

}

// Иначе вычисляем значение внутреннего корня и рекурсивно вызываем эту же функцию для следующего уровня вложенности

return sqrt(1 + (n + k) \* nestedSqrt(n, k + 1));

}

// Функция main является точкой входа в программу

int main() {

// Задаём начальное значение переменной n

int n = 5;

// Вычисляем значение формулы, используя рекурсивную функцию nestedSqrt

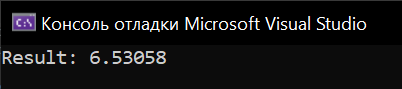
double result = sqrt(1 + (n + 1) \* nestedSqrt(n, 2));

// Выводим результат на экран

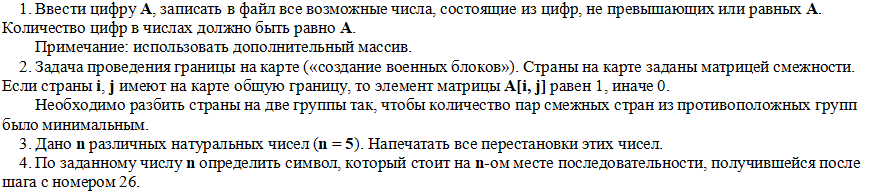
cout << "Result: " << result << endl;

return 0;

}



Дополнительные задания



Задание 1

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

// Рекурсивная функция для генерации чисел из цифр, не превышающих n

void generateNumbers(int n, int len, string prefix, ofstream& outfile) {

// Базовый случай: если мы создали число нужной длины, то записываем его в файл и выходим из рекурсии

if (prefix.length() == len) {

outfile << prefix << endl;

return;

}

// Рекурсивный случай: добавляем все цифры от 0 до n к текущему числу и вызываем функцию для генерации чисел меньшей длины

for (int i = 0; i <= n; i++) {

string newPrefix = prefix + to\_string(i);

generateNumbers(n, len, newPrefix, outfile);

}

}

int main() {

setlocale(LC\_ALL, "Russian");

int a;

cout << "Введите цифру A: ";

cin >> a;

if (a < 0 || a > 9) {

cout << "Ошибка: A должно быть цифрой от 0 до 9" << endl;

return 1;

}

int len;

cout << "Введите длину чисел: ";

cin >> len;

if (len < 1) {

cout << "Ошибка: длина чисел должна быть положительным числом" << endl;

return 1;

}

ofstream outfile("output.txt"); // открываем файл для записи результатов

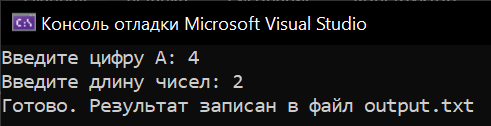
generateNumbers(a, len, "", outfile); // вызываем функцию для генерации чисел

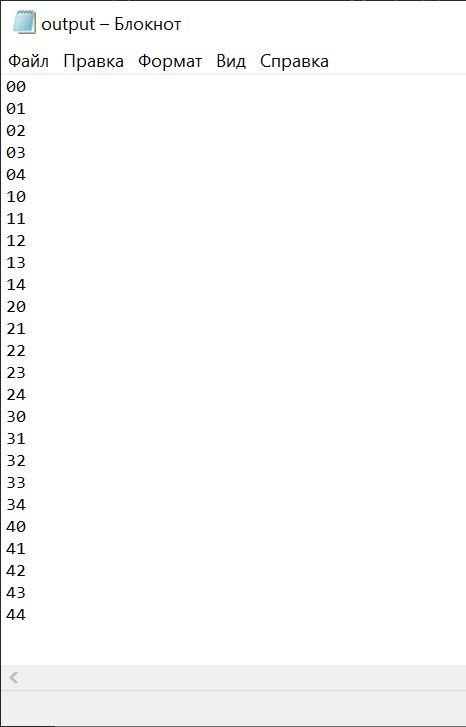
outfile.close(); // закрываем файл

cout << "Готово. Результат записан в файл output.txt" << endl;

return 0;

}





Задание 3

#include <iostream>

using namespace std;

// Функция для печати массива

void printArray(int arr[], int size) {

for (int i = 0; i < size; i++) {

cout << arr[i] << " "; // выводим элементы массива

}

cout << endl; // переходим на новую строку

}

// Рекурсивная функция для нахождения всех перестановок массива

void permute(int arr[], int start, int end) {

// Если начальный индекс равен конечному, значит все элементы обработаны и печатаем массив

if (start == end) {

printArray(arr, end + 1);

return;

}

// Иначе, для каждого элемента, начиная с индекса start до end, вызываем permute(), меняя элементы местами

for (int i = start; i <= end; i++) {

swap(arr[start], arr[i]); // меняем местами текущий элемент и элемент с индексом i

permute(arr, start + 1, end); // вызываем permute() для оставшихся элементов массива

swap(arr[start], arr[i]); // возвращаем массив в исходное состояние

}

}

int main() {

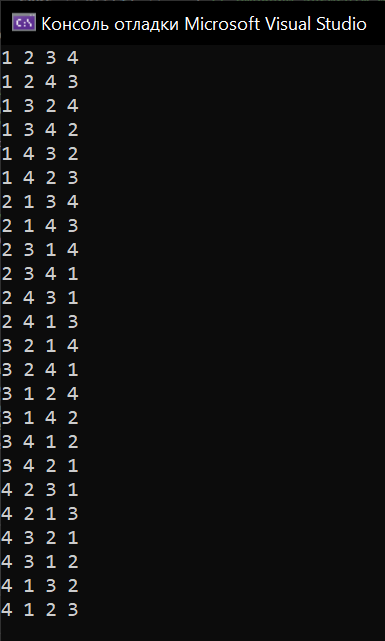
const int n = 4;

int arr[n] = { 1, 2, 3, 4 };

permute(arr, 0, n - 1); // вызываем функцию permute() для перестановок массива

return 0;

}



Задание 4

#include <iostream>

#include <string>

using std::cout; using std::cin; using std::endl;

// Объявление функции findChar

char findChar(char str[], int n);

int main()

{

setlocale(LC\_ALL, "ru");

char str[100];

cout << "Введите строку: ";

gets\_s(str);

cout << endl << str << endl;

int n;

cout << "Введите позицию символа: "; cin >> n;

if (n + 23 > strlen(str))

{

cout << "этого номера нет в строке: ";

}

else

{

// Вызов функции findChar

cout << findChar(str, n);

}

return 0;

}

// Определение функции findChar

char findChar(char str[], int n)

{

if (n + 23 >= strlen(str))

{

cout << "этого номера нет в строке: ";

return '\0';

}

else if (n == 0)

{

// Возвращение символа из строки

return str[n + 23];

}

else

{

// Рекурсивный вызов функции

return findChar(str, n - 1);

}

}

