Федеральное государственное автономное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет», ПНИПУ

ЛАБОРАТОРНАЯ РАБОТА

РЕКУРСИИ

Выполнил: студент группы РИС-23-3б

Артем Владимирович Швецов

Проверила: доцент кафедры ИТАС

Ольга Андреевна Полякова

Пермь 2023

Постановка задачи

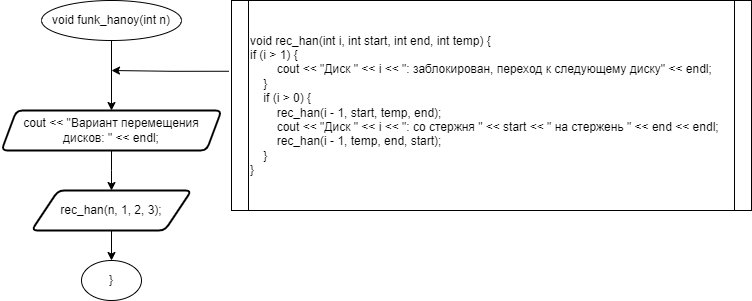
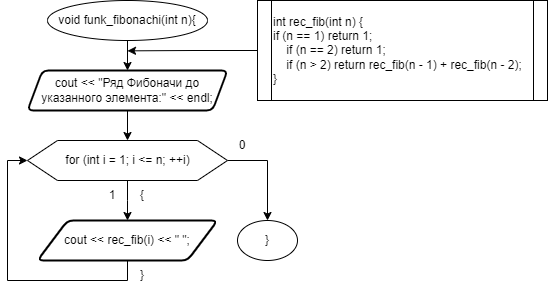
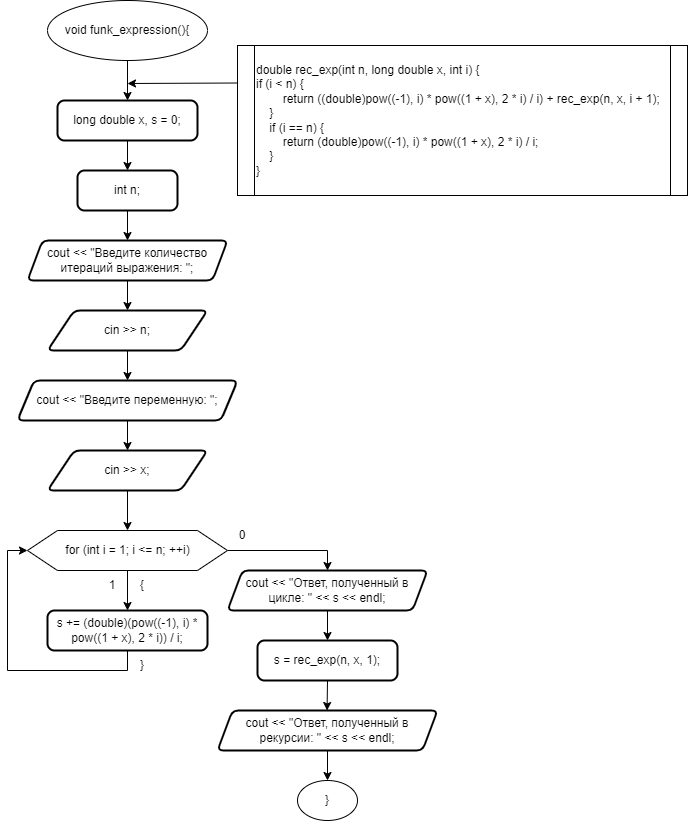
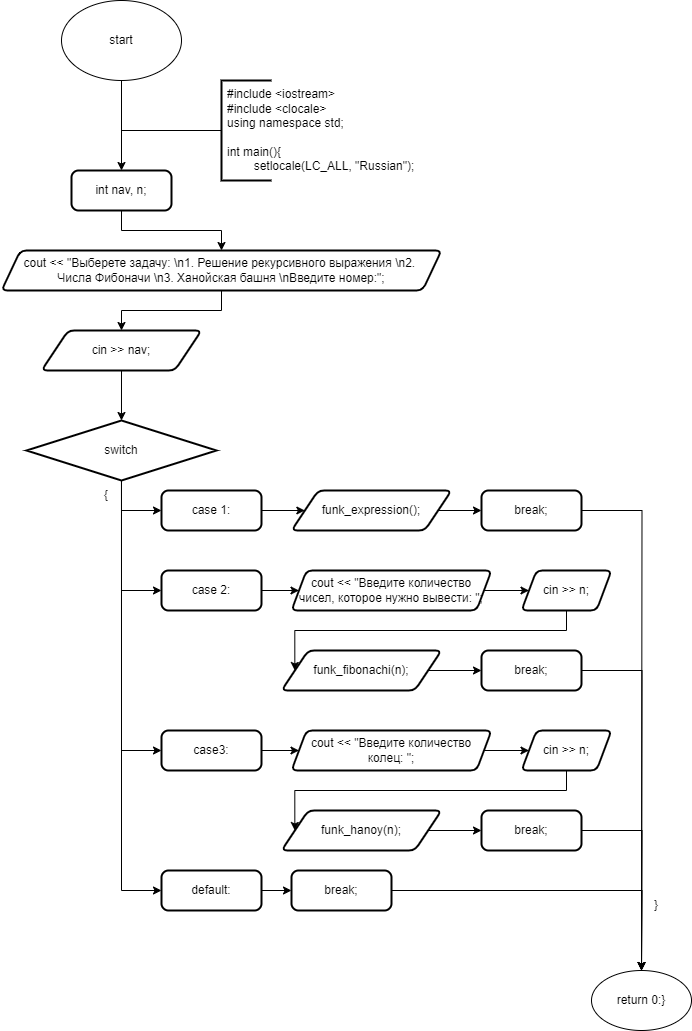
Создать программу для решения трех задач: вывод решения последовательности, вывод ряда Фибоначчи до указанного элемента, вывод решения задачи «Ханойская башня».

Последовательность:

Анализ задачи

1. Каждая из задач решается с использованием рекурсивных функций.
2. При нахождении ряда Фибоначчи рекурсивные функции применяются для определения суммы предыдущих элементов относительно текущего.
3. При решении задачи «Ханойская башня» рекурсивные функции применяются для определения перемещений более мелких, по отношению к текущему, колец.
4. При решении последовательности рекурсивные функции применяются для вычисления суммы элементов, где каждый вызов функции относится к следующему на один шаг от текущего элементу.

Блок-схема



Код

#include <iostream>

#include <clocale>

#include <time.h>

using namespace std;

//В данной программе решение 3-ех задач вызывается через отдельные функции из main()

//Это сделанно для удобства обозрения и редактирования кода

//Рекурсивная функция для вычисления суммы последовательности

double rec\_exp(int n, long double x, int i) {

if (i < n) {

return ((double)pow((-1), i) \* pow((1 + x), 2 \* i) / i) + rec\_exp(n, x, i + 1);

}

if (i == n) {

return (double)pow((-1), i) \* pow((1 + x), 2 \* i) / i;

}

}

//Основная функция для вычисления суммы последовательности

void funk\_expression() {

long double x, s = 0;

int n;

cout << "Введите количество итераций выражения: ";

cin >> n;

cout << "Введите переменную: ";

cin >> x;

auto start\_time\_1 = clock();

//Вычисление через цикл

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

s += (long double)(pow((-1), i) \* pow((1 + x), 2 \* i)) / i;

}

auto end\_time\_1 = clock();

cout << "Ответ, полученный в цикле: " << s << endl;

auto start\_time\_2 = clock();

//Вычисление через рекурсию

s = rec\_exp(n, x, 1);

auto end\_time\_2 = clock();

cout << "Ответ, полученный в рекурсии: " << s << endl;

cout << "Время работы цикла: " << (end\_time\_1 - start\_time\_1) << endl;

cout << "Время работы цикла: " << (end\_time\_2 - start\_time\_2) << endl;

}

//Рекурсивная функция для нахождения ряда Фибоначчи

int rec\_fib(int n) {

if (n == 1) return 1;

if (n == 2) return 1;

if (n > 2) return rec\_fib(n - 1) + rec\_fib(n - 2);

}

//Основная функция для нахождения ряда Фибоначчи

void funk\_fibonachi(int n) {

cout << "Ряд Фибоначи до указанного элемента:" << endl;

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

cout << rec\_fib(i) << " ";

}

}

//Рекурсивная функция для решения "Ханойской башни"

void rec\_han(int i, int start, int end, int temp) {

if (i > 1) {

cout << "Диск " << i << ": заблокирован, переход к следующему диску" << endl;

}

if (i > 0) {

rec\_han(i - 1, start, temp, end);

cout << "Диск " << i << ": cо стержня " << start << " на стержень " << end << endl;

rec\_han(i - 1, temp, end, start);

}

}

//Основная функция для решения "Ханойской башни"

void funk\_hanoy(int n) {

cout << "Вариант перемещения дисков: " << endl;

rec\_han(n, 1, 2, 3);

}

int main() {

setlocale(LC\_ALL, "Russian");

int nav, n;

cout << "Выберете задачу: \n1. Решение рекурсивного выражения \n2. Числа Фибоначи \n3. Ханойская башня \nВведите номер: ";

cin >> nav;

switch (nav) {

case 1:

funk\_expression();

break;

case 2:

cout << "Введите количество чисел, которое нужно вывести: ";

cin >> n;

funk\_fibonachi(n);

break;

case 3:

cout << "Введите количество дисков: ";

cin >> n;

funk\_hanoy(n);

break;

default:

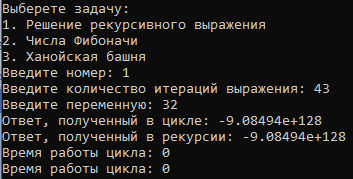
break;

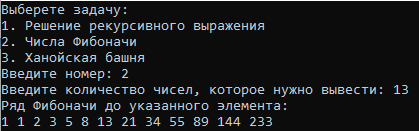
}

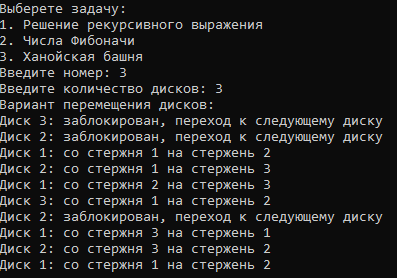
return 0;

}

Решение







Выводы

В ходе работы, были исследованы возможное применение простых рекурсивных функций, что может поспособствовать созданию оптимальных решений в будущем. Однако при определении времени работы программы, значения времени отличались минимально, что указывает на достаточную «легкость» кода при вычислении суммы последовательности и высокую трудность определения точного времени выполнения.

Github

<https://github.com/Hitikov/Lab_Recurtion>