**Звіт**

Автор: Момот Р. КІТ-119а

Дата: 08.04.2020

**ЛАБОРАТОРНА РОБОТА № 2. РЕКУРСИВНІ ТА ІТЕРАЦІЙНІ АЛГОРИТМИ**

**Мета:** Набути навички та практичний досвід у розробці рекурсивних програм.

Індивідуальне завдання

Розробити рекурсивний та ітераційний алгоритми розв’язання індивідуального завдання. Визначити та порівняти час виконання відповідних функцій, зробити висновки.

Індивідуальне завдання: Розробити програму, що визначає кількість n-розрядних двійкових чисел, які не мають у собі підряд двох одиниць.

Текст програми

#include <iostream>

#include <math.h>

#include <stdlib.h>

#include <chrono>

#include <time.h>

using namespace std;

int get\_count\_rekursiya(int n, int prev = 0) {

if (n != 0)

return prev ? get\_count\_rekursiya(n - 1, 0) : get\_count\_rekursiya(n - 1, 0) + get\_count\_rekursiya(n - 1, 1);

else

return 1;

}

int get\_count\_iteratsiya(int size)

{

int\* results = new int[size + 1];

results[0] = 1;

results[1] = 2;

for (int x = 2; x <= size; x++)

results[x] = results[x - 1] + results[x - 2];

return results[size];

}

int main()

{

setlocale(LC\_ALL, "Russian");

int size, result\_rekursiya;

int result\_iteratsiya;

cout << "Введите кол-во разрядов двоичного числа: ";

do{ cin >> size;

} while (size <= 1 && cout << "\nНе может быть меньше 1. Повторите попытку!\nвведите кол-во разрядов двоичного числа: ");

auto start = std::chrono::steady\_clock::now(); //начало замера времени

result\_rekursiya = get\_count\_rekursiya(size); //вызов рекурсии

auto end\_rekursiya = std::chrono::steady\_clock::now(); //замер времени работы рекурсии

auto time\_rekursiya = std::chrono::duration\_cast<std::chrono::nanoseconds>(end\_rekursiya - start); //вычесление времени работы

result\_iteratsiya = get\_count\_iteratsiya(size); //вызов итерации

auto end\_iteratsiya = std::chrono::steady\_clock::now(); //замер времени работы итерации

auto time\_iteratsiya = std::chrono::duration\_cast<std::chrono::nanoseconds>(end\_iteratsiya - end\_rekursiya); //вычесление времени работы

cout << endl << "Количество " << size << "-разрядных двоичных чисел, не содержащих рядом две единицы:" << endl;

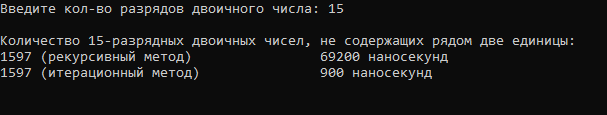
cout << result\_rekursiya << " (рекурсивный метод)\t\t" << time\_rekursiya.count() << " наносекунд" << endl;

cout << result\_iteratsiya << " (итерационный метод)\t\t" << time\_iteratsiya.count() << " наносекунд" << endl;

return 0;

}

Результати роботи програми



Відповіді на питання

1. Який об’єкт називається рекурсивним.

Рекурсивним називається той об’єкт, який частково утворений або визначений за допомогою себе.

1. На що треба звернути особливу увагу при розробці рекурсивного алгоритма?

Необхідно забезпечити закінчення роботи рекурсивного алгоритма (умова, при якому алгоритм зупиниться).

1. Яка функція називається прямо рекурсивною? Наведіть приклад.

Прямо рекурсивною називається та функція, яка в своєму тілі містить виклик самої себе. Наприклад:

int Fact(int n)

{

if (n == 0 || n == 1)

return 1;

else

return n \* Fact(n - 1);

}

1. Яка функція називається косвенно рекурсивною? Наведіть приклад.

Косвенно рекурсивною функцією називається та функція, яка викликається іншою функцією.

int a(int value)

{

cout << value << endl;

if (value == 0)

{

return 0;

}

b(value);

}

void b(int value)

{

cout << value \* value << endl;

a(value - 1);

}

1. Наведіть приклади рекурсивного визначення функцій.

Функція, яка виводить на екран числа від N до -5.

int a(int value)

{

if (value == -5) return 0;

cout << value << endl;

a(value - 1);

}

Функція, яка зводить число x в ступінь y.

int power(long int x, unsigned int y)

{

int d = 0;

if (y == 0)

d = 1;

else if (y == 1)

d = x;

else if (y % 2 == 0)

d = power (x \* x, y/2);

else

d = x \* power(x \* x, y/2);

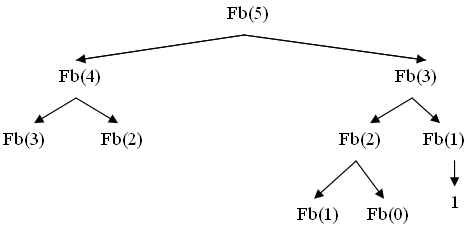
return d;

}

1. В чому полягає потужність рекурсивного визначення?

Потужність рекурсивного визначення об'єкта в тому, що таке кінцеве визначення здатне описувати нескінченно велике число об'єктів.

1. Що таке «дерево рекурсивних викликів»? Наведіть приклади.

Дерево рекурсивних викликів – графічне представлення створенним алгоритмом ланцюжка рекурсивних викликів. Наприклад:

1. Для чого рекурсивні програми потребують додаткову пам’ять у порівнянні із ітераційними програмами?

Рекурсивний варіант додатково витрачає ресурси системи на багаторазові виклики самого себе.

1. Що записується в стек при виклику функції?

При виклику процедури або функції в стек поміщається адреса повернення, стан необхідних регістрів процесора, адреси значень, які повертаються, і передані параметри.

1. Що відбувається, коли функція, яка викликалася, закінчує свою роботу?

Після закінчення роботи функції стек, який зберігає дані, необхідні для роботи функції, очищується.

1. Запишіть ітераційний варіант даної рекурсивної функції:

int FR(int n)

{

if (n == 0) return 1;

return n \* F(n - 1);

}

Ітераційний варіант:

int FR(int n)

{

int fact = 1;

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

fact \*= i;

return fact;

}

1. Запишіть рекурсивний варіант даної ітераційної функції

int FI(int key)

{

for (long i = 0; i < N; i++)

if (m[i] == key) return i;

return -1;

}

Рекурсивний варіант:

int FI(int key, int m[], int value)

{

if (m[value] == key) return value;

else if (value == N) return -1;

else (FI(key, m, value + 1));

}

Висновок

У результаті роботи програми було розроблено програму, яка визначає кількість n-розрядних двійкових чисел, які не мають у собі підряд двох одиниць. Програма використовує 2 варіанти: ітераційний та рекурсивний. Ітераційний метод виконується швидше, бо рекурсивний метод витрачає додаткові ресурси системи при багаторазових викликах самого себе.