Звіт

Автор: Момот Р. КІТ-119а

Дата: 08.04.2020

ЛАБОРАТОРНА РОБОТА № 2. РЕКУРСИВНІ ТА ІТЕРАЦІЙНІ АЛГОРИТМИ

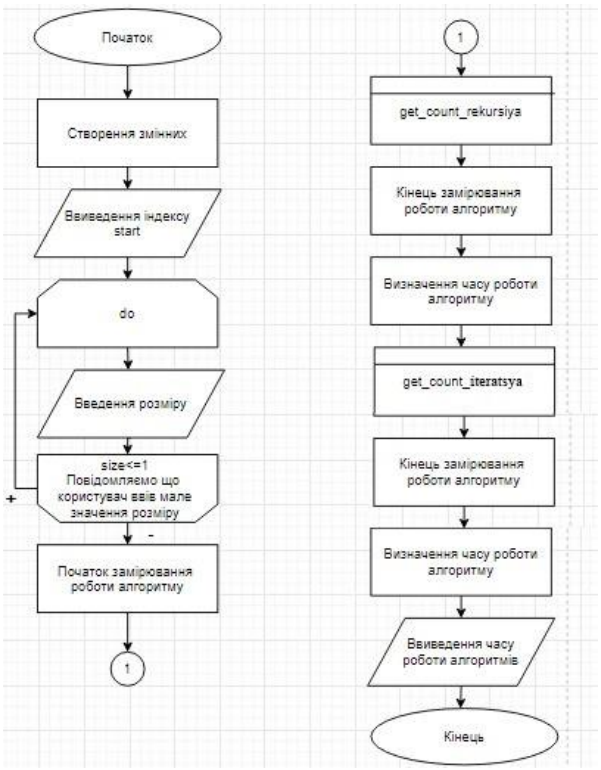
Мета: Набути навички та практичний досвід у розробці рекурсивних програм.

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.

Блок-схема алгоритму програми



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!\nведите 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

```
Введите кол-во разрядов двоичного числа: 15

Количество 15-разрядных двоичных чисел, не содержащих рядом две единицы:
1597 (рекурсивный метод)                69200 наносекунд
1597 (итерационный метод)               900 наносекунд
```

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.

Відповіді на питання

1. Який об’єкт називається рекурсивним.

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

2. На що треба звернути особливу увагу при розробці рекурсивного алгоритму?

Через те, що рекурсивною функцією можна описати нескінчене обчислення, необхідно забезпечити закінчення роботи такої функції.

3. Яка функція називається прямо рекурсивною? Наведіть приклад.

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

```
int Fact(int n)
{
    if (n == 0 || n == 1)
        return 1;
    else
        return n * Fact(n - 1);
}
```

4. Яка функція називається косвенно рекурсивною? Наведіть приклад.

Косвенно рекурсивною функцією називається та функція, яка викликає іншу функцію, яка у свою чергу викликає першу.

```
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);
}
```

5. 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);
```

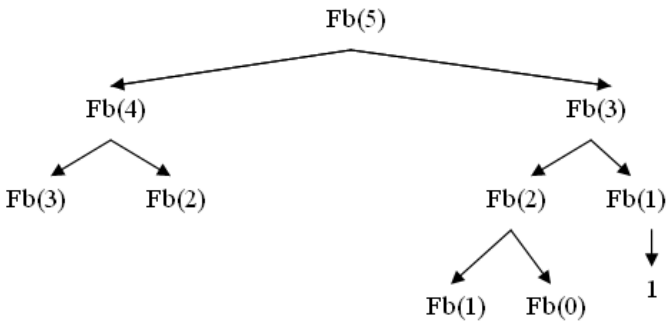
```
        return d;
}
```

6. В чому полягає потужність рекурсивного визначення?

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

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

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



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

Додаткова пам’ять необхідна, тому що при кожному рекурсивному виклику запам’ятовуються попередні значення внутрішніх локальних змінних та переданих у функцію параметрів.

9. 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.

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

При поверненні з кожного рекурсивного виклику старі локальні змінні і параметри виштовхуються зі стеку, читаються значення регістрів і адреса. Тому, виконання програми поновляється від точки виклику метода. 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;
}
```

11.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));
}
```