Лабораторна робота № 1
РЕКУРСІЯ

Мета роботи: отримати навички роботи з рекурсивними алгоритмами.

1.1. Теоретичні відомості

Рекурсія відноситься до одного з фундаментальних понять у математичних та комп'ютерних науках. В мовах програмування рекурсивною програмою називають програму, яка звертається сама до себе (сама себе викликає). Рекурсивна програма не може викликати себе до нескінченності, оскільки у цьому випадку вона ніколи не завершиться. Другою важливою особливістю рекурсивної програми є умова завершення, що дозволяє програмі припинити себе викликати.

Принципові властивості рекурсивного визначення:

1. **визначення об'єкту самого через себе, але з іншими параметрами;**
2. **завершеність ланцюжка визначень на деякому значенні аргументів.**

Необхідність використання рекурсії виникає при реалізації динамічних структур даних, таких як дерева, стеки, черги. Для реалізації рекурсивних алгоритмів передбачена можливість створення рекурсивних функцій. Рекурсивна функція являє собою функцію, у тілі якої здійснюється виклик цієї ж функції.

1.2. Приклади

1. Використання рекурсивної функції для обчислення факторіала.

Нехай потрібно скласти програму для обчислення факторіала будь-якого додатного числа.

Приклад реалізації на С++:

```
#include <iostream>
using namespace std;

int fact(int n);
int main()
{
int m;
cout << "\nВВедіть ціле число:";
cin >> m;
cout << "\n Факторіал числа " << m << " дорівнює " <<
fact (m) ;
```

```
return 0; }

int fact(int n)
{
int a;
if (n<0) return 0;
if (n==0) return 1;
a =n * fact(n-1);
return a;
}
```

Для від'ємного аргументу факторіала не існує, тому функція в цьому випадку повертає нульове значення. Так як факторіал 0 дорівнює 1 за означенням, то в тілі функції передбачений і цей варіант. У випадку коли аргумент функції **fact()** відмінний від 0 та 1, то викликаємо функцію **fact()** із зменшеним на одиницю значенням параметра і результат множиться на значення поточного параметра. Таким чином, у результаті вбудованих викликів функцій повертається наступний результат:

```
n * (n-1) * (n-2) * . . . * 2 * 1 * 1
```

Остання одиниця при формуванні результату належить виклику **fact(0)**.

Розглянемо і іншу реалізацію функції **fact()** для невід'ємного числа n.

```
int fact (int n)
{
return n ? n*fact(n-1):1;
}
```

Приклад реалізації на Python:

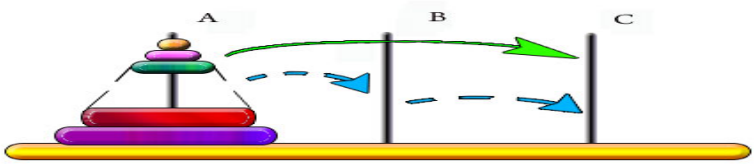
```
n=5
def fact(n):
    if(n==1 or n==0):
        return 1
    else:
        return n*fact(n-1)

print("The factorial of ",n," is: ",fact(n))
```

2. Задача «Ханойські вежі»

Ні один розгляд поняття рекурсії не був би повним без розгляду старовинної задачі про Ханойські вежі.

Принц **Шак'я-Муні** (623 - 544 р.р. до Р.Хр.), якого ще називали **Буддою**, що означає "просвітлений", під час однієї зі своїх подорожей заснував у Ханої (В'єтнам) монастир. У головному храмі монастиря стоїть три стержня

(дерев'яні вісі). На одну з них Будда надягнув **64** дерев'яні диски, усі різного діаметру, причому найширший поклав униз, а решту впорядкував за зменшенням розміру:



Слід переставити піраміду з вісі **A** на вісь **C** у тому ж порядку, користуючись віссю **B**, як допоміжною, та додержуючись таких правил:

a. за один хід переставляти лише один диск з довільної осі на довільну (а не кілька);
b. забороняється класти більший диск на менший, тобто впорядкованість на кожній осі має зберігатися.

Ченці монастиря перекладають, не зупиняючись ні на мить, щосекунди по одному диску і досі. Коли піраміду буде складено на осі С, наступить кінець світу.

Рекурсивний підхід до програмування можна застосувати, якщо розуміти вежу з **n** дисків, як вежу з **(n-1)** диску, що стоїть ще на одному.
Як переставити вежу з двох дисків з **A** на **C**?

1. перекласти диск з **A** на **B**;
2. перекласти диск з **A** на **C**;
3. перекласти диск з **B** на **C**;

Аналогічно програмується повний алгоритм.

Щоб переставити вежу з **n** дисків (позначимо її **P(n)**) з **A** на **C**, слід:

1. **P(n-1)** переставити з **A** на **B**;
2. **P(1)** перекласти з **A** на **C**;
3. **P(n-1)** переставити з **B** на **C**;

1.3. Порядок виконання роботи

1.3.1. Проаналізувати умову задачі.

1.3.2. Розробити алгоритм та створити програму розв’язання задачі згідно з номером варіанту.
1.3.3. Результати роботи оформити протоколом.

1.4. Варіанти завдань

З використанням рекурсії розв’язати наступні задачі.

1. Піднести до додатного цілого степеня дійсне ненульове число.
2. Знайти НСД двох цілих чисел за алгоритмом Евкліда.
3. Числа Фібоначчі f_n обчислюються за формулами $f_0=f_1=1$; $f_n=f_{n-1}+f_{n-2}$ при n=2,3,... Реалізувати функцію, яка за заданим номером *n* обчислюватиме значення f_n.
4. Рекурсивно знайти суму перших n натуральних чисел.
5. Визначити, чи є задане слово паліндромом (слово, яке читається однаково зліва направо і справа наліво).
6. Рекурсивно обчислити суму цифр заданого натурального числа.
7. Визначити, чи є задане число простим за допомогою рекурсії.
8. Обчислити суму арифметичної прогресії за допомогою рекурсії.
9. Визначити, чи є задана послідовність чисел арифметичною прогресією за допомогою рекурсії.
10. Знайти найменше спільне кратне (НСК) двох цілих чисел за допомогою рекурсії.
11. Знайти суму парних або непарних чисел в масиві за допомогою рекурсії.
12. Рекурсивно визначити, чи є задане число ступенем двійки.
13. Знайти суму перших n непарних (або парних) чисел за допомогою рекурсії.
14. Реалізувати алгоритм для розв’язання задачі «Ханойські вежі». Виписати послідовність ходів для перекладання n дисків вежі (n = 2; 3; 4; 5 дисків, використати онлайн гру).

Задача 1. – «задовільно»
Одна задача із 2.-13. – «добре»
Одна задача із 2.-13. плюс 14. – «відмінно»