 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Комп’ютерний практикум №8**

з дисципліни «Алгоритмізація та програмування»

на тему: «Рекурсія»

Варіант №17

**Виконав:**

студент гр. БС-81

Сєров О. В.

**Перевірив:**

доцент каф. БМК

к.т.н. Алхімова С.М.

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2019

**Завдання:**

1. Ознайомитися з теоретичними відомостями створення рекурсивних функцій та їх використання.
2. Побудувати блок-схему алгоритму вирішення задачі відповідно до свого варіанту.
3. Розробити програмний застосунок, що включає реалізацію рекурсивних функцій користувача відповідно до свого варіанту (глобальні змінні не використовувати, функція main має бути призначена тільки для виклику функцій користувача):

***Знайти найбільший загальний дільник (НЗД) чисел М і N, використовуючи метод Ейлера: якщо М ділиться на N то НЗД(N,М) = N, інакше НЗД(N,М) = НЗД(R,N), де R - залишок від ділення М на N.***

1. Скласти і захистити звіт по роботі.

**🞏 Комп’ютерний практикум без зауважень**

**🞏 Комп’ютерний практикум має зауваження:**

**🞏 несвоєчасний захист**

**🞏 присутні зауваження до блок-схеми:**

**🞏 блок-схема не відповідає коду**

**🞏 в блок-схемі присутній код**

**🞏 виконані не за стандартом:**

**🞏 блок умови 🞏 визначений процес (функція)**

**🞏 оператор вибору 🞏 перехід**

**🞏 цикл 🞏 розміри блоків**

**🞏 інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**🞏 присутні зауваження до коду:**

**🞏 задача завдання вирішена хибно**

**🞏 код програми не компілюється**

**🞏 використано глобальні змінні**

**🞏 типи даних визначені хибно**

**🞏 недостатня декомпозиція на функції користувача**

**🞏 функція main містить лише виклик іншої функції**

**🞏 статичні змінні при роботі з масивами**

**🞏 оформлення коду**

**🞏 присутні зайві символи «{» та «}»**

**🞏 інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**🞏 результати виконання програми на рисунках не відповідають коду**

**🞏 невірні відповіді на запитання:**

**🞏 №1 🞏 №2 🞏 №3 🞏 №4 🞏 №5**

**🞏 №6 🞏 №7 🞏 №8 🞏 №9 🞏 №10**

**🞏 незнання теоретичного матеріалу**

**🞏 маються інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Блок-схема:**









**Код:**

#include <iostream>

#include <conio.h>

using namespace std;

//GCD - Greatest common divisor

/\*

функция для нахождения НОД двух чисел рекурсивным методом Эйлера

параметры - 2 отсортированных числа для нахождения наибольшего общего делителя

ничего не возвращает, выводит результат - НОД

\*/

void GCD(int n = 0, int m = 0)

{

if (m % n == 0)

cout << "\nGreatest common divisor is " << n;

else

GCD(m % n, n);

}

/\*

функция для вызова функции GCD с правильными параметрами

параметры - 2 числа для нахождения наибольшего общего делителя

ничего не возвращает

\*/

void right\_comb\_GCD(int a = 0, int b = 0)

{

if (a <= b)

GCD(a, b);

else

GCD(b, a);

}

/\*

функция для корректного ввода числа

параметров нет

возвращает: правильно введенное число

\*/

int input(void)

{

cout << "Enter number to find greatest common divisor\nIt must be bigger than zero\nNumber is : ";

int num = 0;

while (!(cin >> num) || num < 0)

{

cout << "Try correct number!\n";

cin.clear();

cin.ignore();

fflush(stdin);

}

return num;

}

void main()

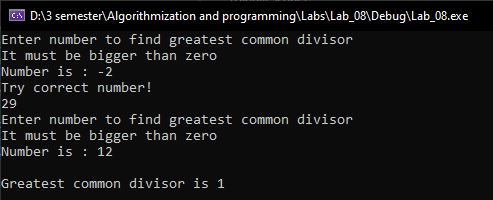
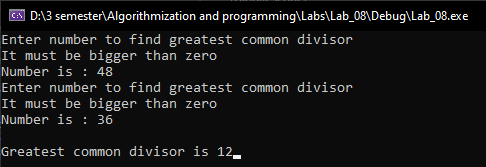
{

right\_comb\_GCD(input(), input());

\_getch();

}

**Результати:**

****

**Контрольні питання:**

1. ***Дайте визначення рекурсії та наведіть приклади її використання в різних галузях знань.***

Рекурсія - процедура в програмуванні, у тілі якої знаходиться явне звернення до неї самої, або через іншу процедуру.

Приклади: у математиці: математична індукція, числа Фібоначчі, факторіал.

Фрактали часто описуються нескінченною рекурсією.

1. ***В яких задачах доцільно використовувати рекурсивні функції?***

Дуже часто під час розробки програм вихідна задача може бути зведена до більш простої. Серед задач може виявитися і початкова, але в спрощеній формі. Наприклад, обчислення функції f(n) може потребувати обчислення f(n -l) і ще якихось операцій; іншими словами, частиною алгоритму обчислення функції буде обчислення цієї ж функції.

1. ***Що спільного та в чому полягає різниця між циклічними та рекурсивними способами організації розрахунків?***

Їх можна використовувати взаємозамінно для вирішення різних проблем. По суті, можна писати рекурсивні функції ітеративно і навпаки.

Ітерація може збільшити продуктивність вашої програми. У той час як рекурсія може дати більш інтуїтивний і елегантний результат. Ви можете вибрати будь-який з них за вашим бажанням.

Ітераційні алгоритми – це висока продуктивність, але важко писати (іноді важко читати)

Рекурсивні алгоритми – це швидкий запис, але погана продуктивність (іноді легше зрозуміти).

1. ***Визначити основні особливості роботи рекурсивних функцій в мові програмування С++.***

У мові програмування C++ із однієї функції можна викликати будь-яку кількість інших функцій. Функції також можуть викликати самі себе, тобто функції мають властивість рекурсивності.

Будь-яка функція в мові C++ має реєнтерабельний (призначений для повторного входу) код, що дозволяє їй звертатися до самої себе безпосередньо або через інші функції. Такі звернення називаються рекурсивними викликами або рекурсією.

1. ***Що таке пряма та непряма (опосередкована) рекурсії?***

Пряма рекурсія, коли функція викликає сама себе, та непряма (опосередкована), коли функція викликає іншу функцію, а та, в свою чергу, потім викликає першу.

1. ***Коли виконується завершення рекурсивних викликів? Що таке рекурсивне зациклювання та до чого воно призводить?***

Рекурсивні виклики завершуються, коли виконується обов’язкова умова закінчення рекурсії.

Рекурсивне зациклювання – рекурсія без умови закінчення, або така рекурсія, у якій ця умова за деяких причин не виконуються. Зациклювання призводить до переповнення стеку пам’яті та дострокового закінчення програми.

1. ***Що таке область дії змінних?***

Область дії представляє частину програми, в межах якої можна використовувати об'єкт. Як правило, область дії обмежується блоком коду, який полягає в фігурні дужки.

1. ***Як необхідно виконати об’яву змінної, щоб доступ до неї був можливим з будь-якої функції поточного файлу програми?***

Необхідно оголосити змінну поза межами всіх функцій – створити глобальну змінну, яку можна використовувати у будь якому місці програми, починаючи від точки її оголошення.

1. ***Як змінюються значення локальних та глобальних змінних при використанні рекурсивних функцій?***

Локальні змінні видаляються з пам’яті ,коли виконання програми повертається назад з функції до основного коду програми. Глобальні змінні зберігають при цьому свої значення і розміщення в пам’яті.

1. ***Визначте головне обмеження при використанні рекурсії.***

Головне обмеження – максимальна глибина рекурсії, яку визначає доступна пам’ять ОЕМ.