МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ

Кафедра системного проектування

**ЗВІТ**про виконання лабораторної роботи №2

“Дослідження базових алгоритмів++”

з дисципліни «Алгоритми та структури даних»

Виконав:

cтудент І курсу

групи ДА-02

Гринчишин Д.Б.

Перевірили:

..

Київ – 2020

**Мета**. Ознайомитись і дослідити базові класичні алгоритми: пошук простих чисел, переведення числа з однієї системи числення в іншу, рекурсивні алгоритми. Набути навичок їх реалізації мовою програмування С/C++, навчитися вимірювати та порівнювати швидкодію алгоритмів.

**Варіант завдань - 2**

**Завдання 1.**

Знайти перші 100 простих чисел.

1) Реалізувати булеву функцію bool isPrime(int n), яка перевіряє вхідне число n на простоту за допомогою звичайного перебору дільників.

2) Реалізувати алгоритм “Решето Ератосфена” для знаходження всіх простих чисел на проміжку від 1 до N.

3) Розв’язати задану задачу у 2 способи: за допомогою функції isPrime() та за допомогою решета Ератосфена, заміряти та вивести час виконання кожного рішення.

1.1 Функція для класифікації числа як просте чи непросте.

#include<time.h>

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#include<stdbool.h>

bool isPrime(int n)

{

int len = floor(sqrt(n));

bool prime = true;

for (int i = 2; i <= len; i++)

{

if (n % i == 0)

{

prime = false;

break;

}

}

return prime;

}

void main() {

int n;

scanf\_s("%d", &n);

printf("%s", isPrime(n) ? "is prime" : "is not prime");

}

1.2 «Решето Ератосфена» для довільного діапазону:

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

void resheto(int len)

{

bool\* primes = new bool[len + 1];

for (int i = 0; i <= len; i++) { primes[i] = true; }

for (int i = 2; i <= ceil(sqrt(len)); i++)

{

if (primes[i])

{

for (int j = pow(i, 2); j <= len; j += i)

{

primes[j] = false;

}

}

}

for (int i = 0; i <= len; i++)

{

if (primes[i])

{

printf("%d\n", i);

}

}

}

void main() {

int diapazon;

printf("diapazon:");

scanf\_s("%d", &diapazon);

resheto(diapazon);

}



1.3 Знаходження перших n = 100 простих чисел «Решетом Ератосфена» та перебором всіх чисел діапазону і порівняння часу виконання цих алгоритмів. #include<time.h>

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#include<locale.h>

#include<stdbool.h>

#include<windows.h>

float resheto(int len)

{

float time\_for\_calc = clock();

bool\* primes = new bool[len + 1];

for (int i = 0; i <= len; i++) { primes[i] = true; }

for (int i = 2; i <= ceil(sqrt(len)); i++)

{

if (primes[i])

{

for (int j = pow(i, 2); j <= len; j += i)

{

primes[j] = false;

}

}

}

for (int i = 2; i <= len; i++)

{

if (primes[i])

{

printf("%d ", i);

}

}

time\_for\_calc = clock() - time\_for\_calc;

return time\_for\_calc;

}

bool isPrime(int n)

{

int len = floor(sqrt(n));

bool prime = true;

for (int i = 2; i <= len; i++)

{

if (n % i == 0)

{

prime = false;

break;

}

}

return prime;

}

float perebor(int nb\_primes, int n = 2, int i = 0)

{

float time\_for\_calc = clock();

while(i < nb\_primes)

{

if (isPrime(n))

{

i++;

printf("%d ", n);

}

n++;

}

printf("\n\n");

time\_for\_calc = clock() - time\_for\_calc;

return time\_for\_calc;

}

void main()

{

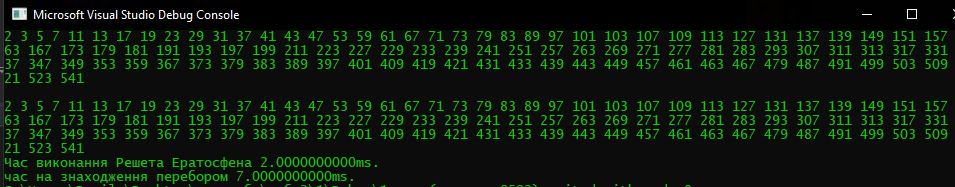
SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int number\_of\_primes = 100;

printf("\nЧас виконання Решета Ератосфена %.10fms.\nчас на знаходження перебором %.10fms.", resheto(542), perebor(number\_of\_primes));

}

**Завдання 2.**

Написати функцію для переведення натурального десяткового числа в 6-ву систему числення, і функцію для зворотнього переводу. Протестувати на декількох прикладах.

**Код програми.**

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#include<windows.h>

int to\_6(int n10, int n6 = 0, int i = 0)

{

while (n10 > 0)

{

n6 += (n10 % 6) \* pow(10, i);

n10 = (n10 - n10 % 6) / 6;

i++;

}

return n6;

}

int back\_to\_10(int n6, int n10 = 0, int i = 0)

{

while (n6 > 0)

{

n10 += (n6 % 10) \* pow(6, i);

n6 = (n6 - n6 % 10) / 10;

i++;

}

return n10;

}

int main()

{

int n;

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

printf("10-ове число: ");

scanf\_s("%d", &n);

printf("В 6-овій системі числення: %d\n", to\_6(n));

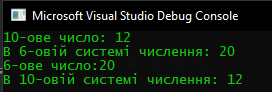
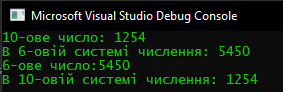
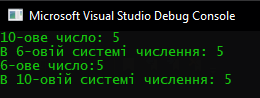
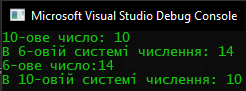
printf("6-ове число:");

scanf\_s("%d", &n);

printf("В 10-овій системі числення: %d\n", back\_to\_10(n));

return 1;

}

**Завдання 3.**

Порахувати кількість чисел у послідовності Фібоначчі, менших заданого натурального числа N, за допомогою рекурсії.

**Код програми.**

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#include<stdbool.h>

#include<windows.h>

int func(int last1, int last2, int i, int limit)

{

int temp = last2 + last1;

if (temp < limit)

{

last1 = last2;

last2 = temp;

i++;

func(last1, last2, i, limit);

}

else return i;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int last1 = 0, last2 = 1, i = 2, limit;

printf("Введіть ліміт: ");

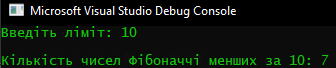
scanf\_s("%d", &limit);

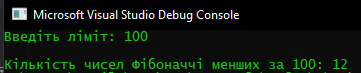
i = func(last1, last2, i, limit);

printf("\nКількість чисел Фібоначчі менших за %d: %d", limit, i);

return 1;

}





**Висновок:**

Я здобув конкретні навички та знання в написанні коду на мові програмування C, а саме набув навичок в реалізації таких класичних алгоритмів, як: пошук простих чисел, переведення числа з однієї системи числення в іншу, рекурсивні алгоритми пошуку чисел з ряду Фібоначчі. Навчився заміряти час виконання алгоритмів та порівнювати їх складність.

**Контрольні запитання:**

1) На різницю в часі між виконанням алгоритмів знаходження простих чисел перебором дільників і «Решетом Ератосфена» впливає те, що алгоритм перебору дільників перебирає всі числа, що менші за корінь потенціального простого числа, в цей же час «Решето Ератосфена» перебирає лише прості дільники, що дозволяє алгоритму виконати нелінійно меншу кількість операцій.

2) Складність переведення числа 0,3 в двійкову систему числення в тому, що воно перетворюється в нескінченний ірраціональний дріб.

3)Рекурсія та цикли набувають конкретних переваг в окремих ситуаціях і їх використання залежить від поставленої задачі та її специфіки. Загалом, основною перевагою рекурсії є короткість і зрозумілість запису та зручність при редагуванні. Основним мінусом є нераціональність в плані використання обчислювальних ресурсів, тому більшість рекурсивних алгоритмів, що виконуються за нерозумну кількість часу, при певних вхідних значеннях, при перетворенні в ітеративні стануть менш затратними, як наприклад знаходження чисел з ряду Фібоначчі.

.