**Київський національний університет імена Тараса Шевченка  
Факультет комп’ютерних наук та кібернетики**

**Алгоритми та складність  
Лабораторна робота №7**

**Алгоритм Рабіна-Карпа для знаходження підматриці**

**Звіт**

**Підготував:**студент групи К-29  
Дацюк Віталій Олегович

**Київ-2019**

1. **Постановка завдання.**

Узагальніть метод Рабіна-Карпа пошуку зразка в текстовому рядку так, щоб він дозволив розв’язати задачу пошуку заданого зразка розміром *m* на *m* у символьному масиві розміром *n* на *n*. Зразок можна рухати по горизонталі та вертикалі, але не обертати.

1. **Опис алгоритму Рабіна-Карпа**

Для простоти припустимо, що алфавіт складається з десяткових цифр Σ = {0,1,…,9}. (В загальному випадку можна припустити, що кожний символ — це цифра в системі числення з основою d, де d = |Σ|.) Після цього, рядок з k символів, можна розглядати як число довжини k. Тобто символьний рядок «12345» відповідає числу 12345.

Для заданого зразка P[1..m] позначимо через p відповідне йому десяткове значення. Аналогічно, для заданого тексту T[1..n] позначимо через ts десяткове значення підрядка T[s+1..s+m] довжини m при s = 0,1,…,n-m. Очевидно, що ts=p тоді і тільки тоді, коли T[s+1..s+m]=P[1..m]; таким чином, s — допустимий зсув тоді і тільки тоді, коли ts=p.

Якщо значення p можна обчислити за Θ(m) а значення ts за сумарний час Θ(n-m+1), то усі допустимі зсуви можна було б знайти за час Θ(m) + Θ(n-m+1) = Θ(n) шляхом порівняння p з кожним з можливих ts. (Покищо до уваги не береться той факт, що величини p і ts можуть виявитись дуже великими.)

З допомогою схеми Горнера величину p можна обчислити за час Θ(m):

p=P[m]+10(P[m-1]+10(P[m-2]+…+10(P[2]+10P[1]))…)).

Значення t0 можна обчислити з масиву T[1..n] аналогічним способом за час Θ(m). В той же час, знаючи величину ts величину ts+1 можна обчислити за фіксований час:

ts+1 =10(ts -10m-1\*T[s+1])+T[s+m+1].

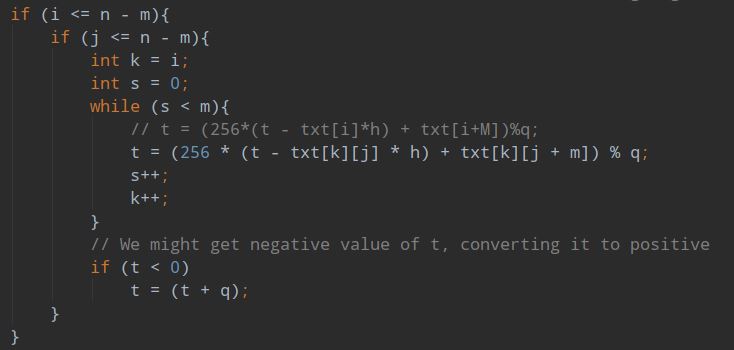
**Алгоритм полягає в наступному:**

* обчислити число p;
* обчислити всі ts;
* Для тих s для яких ts =p, виконати перевірку P[1..m] = T[s+1..s+m].

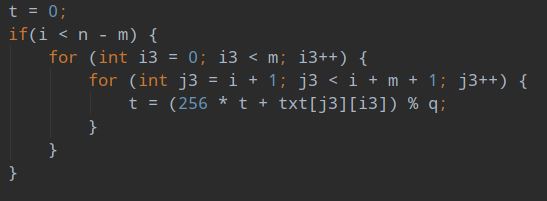
**Складність:** O(n+m)

1. **Опис алгоритму Рабіна-Карпа для знаходження підматриці**

Розроблюючи поданий алгоритм я зберіг головну ідею звичайного алгоритму Рабіна-Карпа. Відмінність полягає утому, що я вираховую початковий хеш не для строки, а для першої підматриці розміром m\*m(m – довжина і ширина патерну. Вважаємо, що m < n. Початок першої підматриці в індексах i = 0;j=0). Тоді я так само порівнюю хеші. Якщо вони рівні то перевіряю посимвольно, інакше я вираховую наступний хеш по такій схемі:



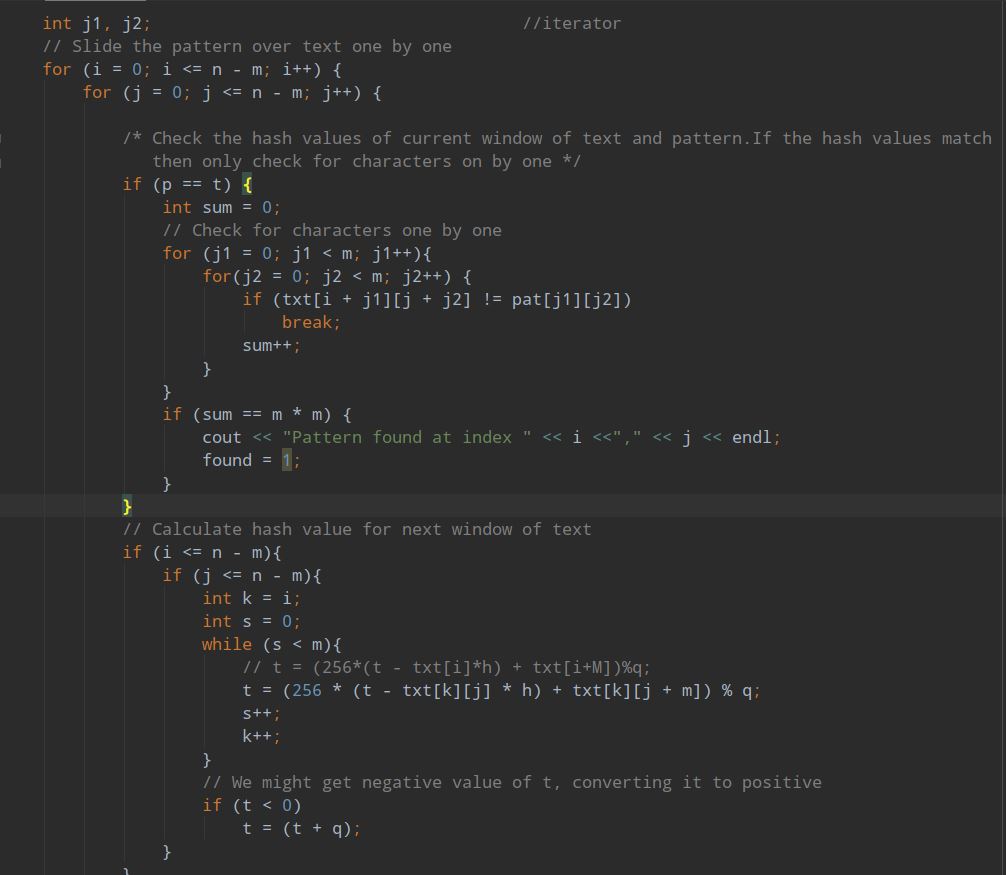
Коли я дійшов до краю матриці, я заново вираховую новий хеш для підматриці m\*m, початок якої - клітінка з індексом 1,0 (якщо цього дозволяє розмір матриці n\*n)



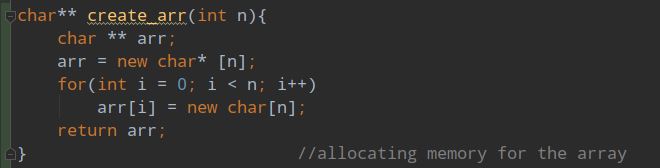
І повторюю дії, описані вище, доки нам буде дозволяти того розмір матриці.

1. **Основні модулі програми**

* **Алгоритм Рабіна-Карпа**

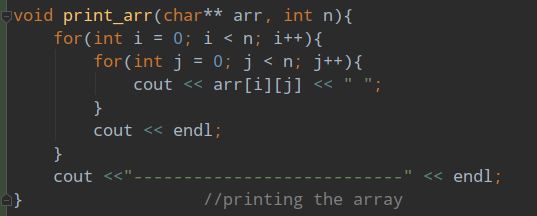


* **Виділення пам’яті для масиву**

****

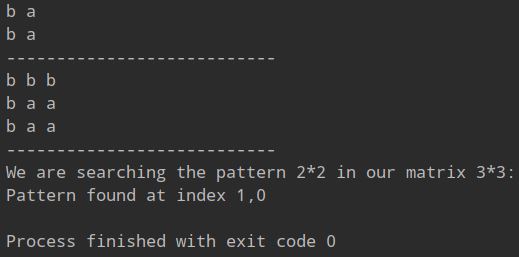
1. **Інтерфейс користувача**

* **Функція виводу матриці на екран**

****

Також до інтерфейсу належать всі функції, які були описані вище.

1. **Тестові приклади**



1. **Література**

* <https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%A0%D0%B0%D0%B1%D0%B8%D0%BD%D0%B0_%E2%80%94_%D0%9A%D0%B0%D1%80%D0%BF%D0%B0>
* <https://www.geeksforgeeks.org/rabin-karp-algorithm-for-pattern-searching/><https://habr.com/ru/post/111449/>