Київський національний університет імені Тараса Шевченка
Факультет комп’ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем
Математичні основи захисту інформації

Лабораторна робота №9
“Побудова кілець”

Виконали студенти 3-го курсу
Групи ІПС-32

Роботу виконали:
Ольховатий Ігор
Ковальов Володимир
Тряско Софія
Цілинко Олександр
Бондар Юлія
Волик Артем

Київ 2023

Завдання:

Побудувати кільце використовуючи 2 алгоритму. На вхід подається визначальний рядок і порядок кільця

Визначальний рядок

$(1, 7, 2, 5, 11, 13, 17, 19, 3, 4, 6, 12, 8, 10, 9, 14, 18, 16, 15, 0)$

Підстановка

7->2
2->5
5->11
11->13
13->17
17->19
19->3
3->4
4->6
6->12
12->8
8->10
10->9
9->14
14->18
18->16
16->15
15->0

1 7 5
4 6 11
12 2 10
14 9 13
8 17 18
0 15 19
16 3

Побудуємо таблицю додавання

На першому кроці таблиця має такий вигляд(0 заповнені елементи, які ще не пораховані)

```
    0     1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17    18    19
    1     7     5     4     6    11    12     2    10    14     9    13     8    17    18     0    15    19    16     3
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
```

Запишемо приклад, як буде працювати для $c = 1\, c` = 1 + 1 = 7$
Отже $c` + x = (c + 1) + x = c + (1 + x)$
$x = 0 \Rightarrow c` + 0 = c + 1 = 1 + 1 = 7 \Rightarrow 7 + 0 = 7$
$7 + 2 = (1 + 1) + 2 = 1 + (1 + 2) = 1 + 2 = 2$
і так далі
Зробимо цей алгоритм на мові C++
Маємо наступну таблицю

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	7	5	4	6	11	12	2	10	14	9	13	8	17	18	0	15	19	16	3
2	5	13	12	8	17	10	11	14	16	18	19	9	3	15	7	1	4	0	6
3	4	12	16	15	8	0	6	7	5	2	10	1	9	11	19	17	14	13	18
4	6	8	15	0	10	1	12	2	11	5	9	7	14	13	3	19	18	17	16
5	11	17	8	10	19	9	13	18	15	16	3	14	4	0	2	7	6	1	12
6	12	10	0	1	9	7	8	5	13	11	14	2	18	17	4	3	16	19	15
7	2	11	6	12	13	8	5	9	18	14	17	10	19	16	1	0	3	15	4
8	10	14	7	2	18	5	9	13	19	17	16	11	15	3	12	6	0	4	1
9	14	16	5	11	15	13	18	19	4	3	0	17	1	6	10	8	7	12	2
10	9	18	2	5	16	11	14	17	3	19	15	13	0	4	8	12	1	6	7
11	13	19	10	9	3	14	17	16	0	15	4	18	6	1	5	2	12	7	8
12	8	9	1	7	14	2	10	11	17	13	18	5	16	19	6	4	15	3	0
13	17	3	9	14	4	18	19	15	1	0	6	16	12	7	11	5	8	2	10
14	18	15	11	13	0	17	16	3	6	4	1	19	7	12	9	10	2	8	5
15	0	7	19	3	2	4	1	12	10	8	5	6	11	9	16	18	13	14	17
16	15	1	17	19	7	3	0	6	8	12	2	4	5	10	18	14	11	9	13
17	19	4	14	18	6	16	3	0	7	1	12	15	8	2	13	11	10	5	9
18	16	0	13	17	1	19	15	4	12	6	7	3	2	8	14	9	5	10	11
19	3	6	18	16	12	15	4	1	2	7	8	0	10	5	17	13	9	11	14

Бачимо, що наші розрахунки збігаються з даними з таблиці, а отже алгорит закодовано правильно
Запишемо приклад як такий алгоритм буде працювати для множення
$c = 1\, c` = 1 + 1 = 7$
Отже $c` * x = (c + 1) * x = c * x + x$
$x = 0 \Rightarrow c` * 0 = c * 0 + 1 * 0 = 0 + 0 = 0 \Rightarrow 7 * 0 = 0$
$7 * 2 = (1 + 1) * 2 = 1 * 2 + 2 = 2 + 2 = 13$

Тепер зробимо теж саме з таблицею для множення

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	2	3	17	4	12	8	13	15	11	7	9	14	16	19	18	10	1	6	5
0	3	17	1	4	14	15	16	18	9	13	11	19	10	5	6	7	2	8	12
0	4	4	4	0	0	4	0	4	4	0	4	0	0	0	4	0	4	4	0
0	5	12	14	0	14	5	19	12	0	14	0	19	5	5	14	12	19	19	12
0	6	8	15	4	5	1	7	2	11	10	9	12	13	14	3	16	18	17	19
0	7	13	16	0	19	7	5	13	4	19	4	5	12	12	16	14	10	10	14
0	8	15	18	4	12	2	13	3	9	7	11	14	16	19	17	10	6	1	5
0	9	11	9	4	0	11	4	9	11	4	9	0	4	0	11	4	11	9	0
0	10	7	13	0	14	10	19	7	4	14	4	19	5	5	13	12	16	16	12
0	11	9	11	4	0	9	4	11	9	4	11	0	4	0	9	4	9	11	0
0	12	14	19	0	19	12	5	14	0	19	0	5	12	12	19	14	5	5	14
0	13	16	10	0	5	13	12	16	4	5	4	12	14	14	10	19	7	7	19
0	14	19	5	0	5	14	12	19	0	5	0	12	14	14	5	19	12	12	19
0	15	18	6	4	14	3	16	17	11	13	9	19	10	5	1	7	8	2	12
0	16	10	7	0	12	16	14	10	4	12	4	14	19	19	7	5	13	13	5
0	17	1	2	4	19	18	10	6	11	16	9	5	7	12	8	13	3	15	14
0	18	6	8	4	19	17	10	1	9	16	11	5	7	12	2	13	15	3	14
0	19	5	12	0	12	19	14	5	0	12	0	14	19	19	12	5	14	14	5

І дійсно в таблиці маємо відповідні значення

Програмна реалізація:

```
#include <iostream>
#include <iomanip>

void print(int** arr, int size) {
        for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
```

```
		std::cout << std::setw(5)<< arr[i][j] << " ";
		}
		std::cout << std::endl;
		}
}
void foo(int** arr) {
		int c = 1;
		int c_temp = arr[1][c];
		int count = 0;
		while (c_temp!=0) {
		for (int i = 0; i < 20; i++) {
		arr[c_temp][i] = arr[1][arr[c][i]];
		}
		c = c_temp;
		c_temp = arr[1][c];
		count++;
		}
}
void multi(int** arr,int** add) {
		int c = 1;
		int c_temp = add[1][c];
		int count = 0;
		while (c_temp != 0) {
		for (int i = 0; i < 20; i++) {
		arr[c_temp][i] = add[i][arr[c][i]];
		}
		c = c_temp;
		c_temp = add[1][c];
		count++;
		}
}
int main()
{
		//int a[20] = { 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19 };
		int a[20] = { 1,7,5,4,6,11,12,2,10,14,9,13,8,17,18,0,15,19,16,3 };
		int** arr = new int* [20];
		for (int i = 0; i < 20; i++) {
		arr[i] = new int[20];
		}
		for (int i = 0; i < 20; i++) {

		for (int j = 0; j < 20; j++) {

		arr[i][j] = 0;
		}
		}
		for (int i = 0; i < 20; i++) {
		arr[0][i] = i;
```

```
        }
        for (int i = 0; i < 20; i++) {
        arr[1][i] = a[i];
        }

        //print(arr, 20);

        foo(arr);
        std::cout << "------------------------\n";
        //print(arr, 20);

        int** mult = new int* [20];
        for (int i = 0; i < 20; i++) {
        mult[i] = new int[20];
        }
        for (int i = 0; i < 20; i++) {

        for (int j = 0; j < 20; j++) {

        mult[i][j] = 0;
        }
        }
        for (int i = 0; i < 20; i++) {
        mult[0][i] = 0;
        }
        for (int i = 0; i < 20; i++) {
        mult[1][i] = i;
        }
        multi(mult, arr);
        std::cout << "------------------------\n";

        print(mult,20);
}
```

Література:

- Лекції з предмету “Математичні основи захисту інформації”
- https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D0%BE%D0%B5_%D0%BA%D0%BE%D0%BB%D1%8C%D1%86%D0%BE
- https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D1%83%D1%82%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B5_%D0%BA%D1%96%D0%BB%D1%8C%D1%86%D0%B5