Харківський національний університет імені В.Н. Каразіна
Навчально-науковий інститут комп’ютерних наук та штучного інтелекту

ЗВІТ

З ПРАКТИЧНОЇ РОБОТИ №12

дисципліна: «Алгоритмізація та програмування»

Виконав: студент 2 курсу групи КС22
Спеціальності 122 «Комп’ютерні науки»
Скрипняк Тарас Артемович
Прийняв: викладач
Олешко О.І.

Харків – 2024

Завдання №1: - згенерувати масив 100 випадкових чисел, що не повторюються, з діапазону 1-99999
- порахувати та вивести кількість колізій одержуваних методом ділення при максимальній кількості хешкодів:
- 10 (0-9)
- 100 (0-99)
- для першого випадку реалізувати хеш-таблицю (таблиця, де key – цифра, value – її подання у вигляді рядку) з вирішенням колізій методом ланцюгів.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

typedef struct Node {
    int value;
    struct Node *next;
} Node;

void insert(Node **table, int hash, int value) {
    Node *newNode = (Node *)malloc(sizeof(Node));
    newNode->value = value;
    newNode->next = table[hash];
    table[hash] = newNode;
}

int countCollisions(Node **table, int tableSize) {
    int collisions = 0;
    for (int i = 0; i < tableSize; i++) {
        int count = 0;
        Node *current = table[i];
        while (current) {
            count++;
            current = current->next;
        }
        if (count > 1)
            collisions += count - 1;
    }
    return collisions;
}

void freeTable(Node **table, int tableSize) {
    for (int i = 0; i < tableSize; i++) {
        Node *current = table[i];
        while (current) {
            Node *temp = current;
            current = current->next;
            free(temp);
        }
    }
}

void fillArrayRand(int *array, int size, int max) {
    int used[max + 1];
    for (int i = 0; i <= max; i++)
        used[i] = 0;

    int count = 0;
```

```
    while (count < size) {
        int num = rand() % max + 1;
        if (!used[num]) {
            used[num] = 1;
            array[count++] = num;
        }
    }
}

void printHashTable(Node **table, int tableSize) {
    printf("\nTable:\n");
    for (int i = 0; i < tableSize; i++) {
        if (table[i] != NULL) {
            printf("Key %d: ", i);
            Node *current = table[i];
            while (current) {
                printf("%d -> ", current->value);
                current = current->next;
            }
            printf("NULL\n");
        }
    }
}

int main() {
    srand(time(0));

    int numbers[100];
    fillArrayRand(numbers, 100, 99999);
    int tableSizes[] = {10, 100};
    for (int i = 0; i < 2; i++) {
        int tableSize = tableSizes[i];
        Node *hashTable[tableSize];
        for (int j = 0; j < tableSize; j++)
            hashTable[j] = 0;

        for (int j = 0; j < 100; j++)
            insert(hashTable, numbers[j] % tableSize, numbers[j]);

        printHashTable(hashTable, tableSize);
        printf("Collisions for table size %d: %d\n", tableSize,
countCollisions(hashTable, tableSize));
        freeTable(hashTable, tableSize);
    }

    return 0;
}
```

Лістинг - вихідний код програми

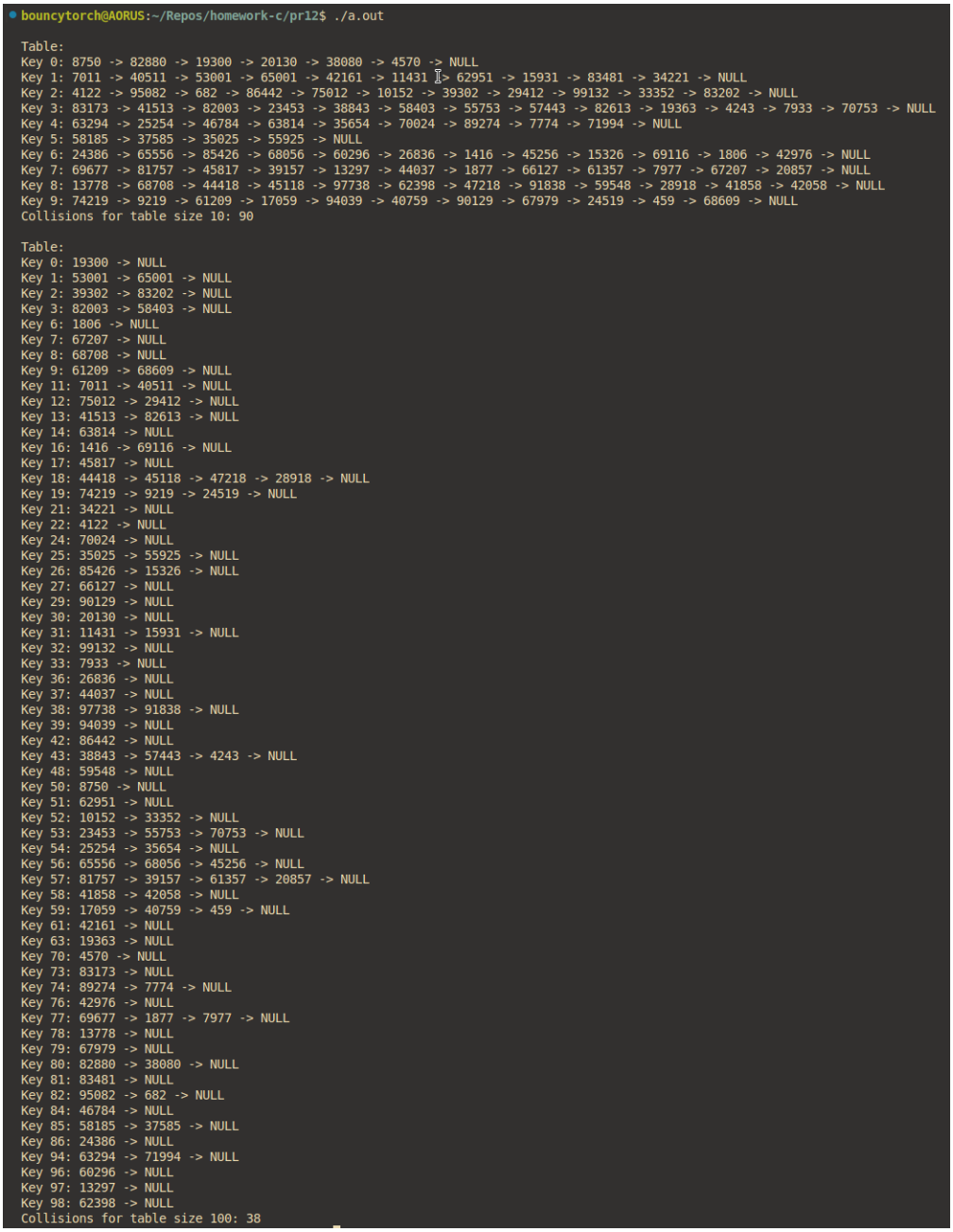

Рисунок 1 - результат виконання програми

Завдання №2: - порахувати та вивести кількість колізій одержуваних методом множення при максимальній кількості хешкодів:
- 10 (0-9)
- 100 (0-99)
- для першого випадку реалізувати хеш-таблицю (таблиця, де key – цифра, value – її подання у вигляді рядку) з вирішенням колізій лінійним методом.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define A 0.618033
```

```
// Функція для генерації унікальних випадкових чисел
void generateUniqueArray(int *arr, int n, int max_val) {
    int i = 0;
    while (i < n) {
        int num = rand() % max_val;
        int is_unique = 1;
        for (int j = 0; j < i; j++)
            if (arr[j] == num) {
                is_unique = 0;
                break;
            }
        if (is_unique) arr[i++] = num;
    }
}

int hashMultiply(int key, int m) {
    return (int)(m * (key * A - (int)(key * A)));
}

int countCollisions(int *arr, int n, int m) {
    int collisions = 0;
    int hashTable[m];
    for (int i = 0; i < m; i++) hashTable[i] = -1;
    for (int i = 0; i < n; i++) {
        int hash = hashMultiply(arr[i], m);
        if (hashTable[hash] != -1)  collisions++;
        hashTable[hash] = arr[i];
    }
    return collisions;
}

void createHashTable(int *arr, int n, int m) {
    char *hashTable[m];
    for (int i = 0; i < m; i++)
        hashTable[i] = NULL;

    for (int i = 0; i < n; i++) {
        int hash = hashMultiply(arr[i], m);
        int attempts = 0;

        while (hashTable[hash] != NULL && attempts < m) {
            hash = (hash + 1) % m;
            attempts++;
        }

        if (attempts == m) {
            printf("Хеш-таблиця переповнена, неможливо вставити %d\n", arr[i]);
            continue;
        }

        char *value = (char *)malloc(12 * sizeof(char));
        sprintf(value, "%d", arr[i]);
        hashTable[hash] = value;
    }

    for (int i = 0; i < m; i++)
        if (hashTable[i] != NULL)
            printf("key: %d, value: %s\n", i, hashTable[i]);

    for (int i = 0; i < m; i++)
        if (hashTable[i] != NULL)
            free(hashTable[i]);
}


int main() {
```

```
    srand(time(0));

    int numbers[100];
    generateUniqueArray(numbers, 100, 100000);
    printf("Collisions for table size 10: %d\n", countCollisions(numbers, 100,
10));
    printf("Collisions for table size 100: %d\n", countCollisions(numbers, 100,
100));

    createHashTable(numbers, 100, 10);
    createHashTable(numbers, 100, 100);

    return 0;
}
```

Лістинг - вихідний код програми

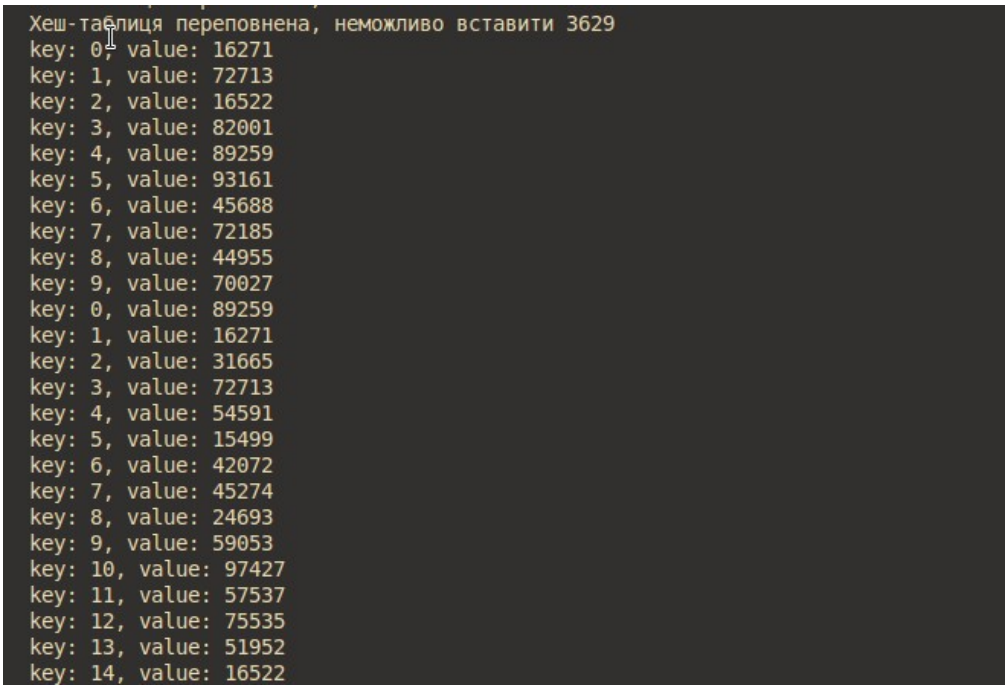

Рисунок 1 - результат виконання програми