

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.

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

```

bouncytorch@AORUS:~/Repos/homework-c/pr12$ ./a.out

Table:
Key 0: 8750 -> 82880 -> 19300 -> 20130 -> 38080 -> 4570 -> NULL
Key 1: 7011 -> 40511 -> 53001 -> 65001 -> 42161 -> 11431 -> 62951 -> 15931 -> 83481 -> 34221 -> NULL
Key 2: 4122 -> 95082 -> 682 -> 86442 -> 75012 -> 10152 -> 39302 -> 29412 -> 99132 -> 33352 -> 83202 -> NULL
Key 3: 83173 -> 41513 -> 82003 -> 23453 -> 38843 -> 58403 -> 55753 -> 57443 -> 82613 -> 19363 -> 4243 -> 7933 -> 70753 -> NULL
Key 4: 63294 -> 25254 -> 46784 -> 63814 -> 35654 -> 70024 -> 89274 -> 7774 -> 71994 -> NULL
Key 5: 58185 -> 37585 -> 35025 -> 55925 -> NULL
Key 6: 24386 -> 65556 -> 85426 -> 68056 -> 60296 -> 26836 -> 1416 -> 45256 -> 15326 -> 69116 -> 1886 -> 42976 -> NULL
Key 7: 69677 -> 81757 -> 45817 -> 39157 -> 13297 -> 44037 -> 1877 -> 66127 -> 61357 -> 7977 -> 67207 -> 20857 -> NULL
Key 8: 13778 -> 68708 -> 44418 -> 45118 -> 97738 -> 62398 -> 47218 -> 91838 -> 59548 -> 28918 -> 41858 -> 42058 -> NULL
Key 9: 74219 -> 9219 -> 61209 -> 17059 -> 94039 -> 40759 -> 90129 -> 67979 -> 24519 -> 459 -> 68609 -> NULL
Collisions for table size 10: 90

Table:
Key 0: 19300 -> NULL
Key 1: 53001 -> 65001 -> NULL
Key 2: 39302 -> 83202 -> NULL
Key 3: 82003 -> 58403 -> NULL
Key 6: 1806 -> NULL
Key 7: 67207 -> NULL
Key 8: 68708 -> NULL
Key 9: 61209 -> 68609 -> NULL
Key 11: 7011 -> 40511 -> NULL
Key 12: 75012 -> 29412 -> NULL
Key 13: 41513 -> 82613 -> NULL
Key 14: 63814 -> NULL
Key 16: 1416 -> 69116 -> NULL
Key 17: 45817 -> NULL
Key 18: 44418 -> 45118 -> 47218 -> 28918 -> NULL
Key 19: 74219 -> 9219 -> 24519 -> NULL
Key 21: 34221 -> NULL
Key 22: 4122 -> NULL
Key 24: 70024 -> NULL
Key 25: 35025 -> 55925 -> NULL
Key 26: 85426 -> 15326 -> NULL
Key 27: 66127 -> NULL
Key 29: 90129 -> NULL
Key 30: 20130 -> NULL
Key 31: 11431 -> 15931 -> NULL
Key 32: 99132 -> NULL
Key 33: 7933 -> NULL
Key 36: 26836 -> NULL
Key 37: 44037 -> NULL
Key 38: 97738 -> 91838 -> NULL
Key 39: 94039 -> NULL
Key 42: 86442 -> NULL
Key 43: 38843 -> 57443 -> 4243 -> NULL
Key 48: 59548 -> NULL
Key 50: 8750 -> NULL
Key 51: 62951 -> NULL
Key 52: 10152 -> 33352 -> NULL
Key 53: 23453 -> 55753 -> 70753 -> NULL
Key 54: 25254 -> 35654 -> NULL
Key 56: 65556 -> 68056 -> 45256 -> NULL
Key 57: 81757 -> 39157 -> 61357 -> 20857 -> NULL
Key 58: 41858 -> 42058 -> NULL
Key 59: 17059 -> 40759 -> 459 -> NULL
Key 61: 42161 -> NULL
Key 63: 19363 -> NULL
Key 70: 4570 -> NULL
Key 73: 83173 -> NULL
Key 74: 89274 -> 7774 -> NULL
Key 76: 42976 -> NULL
Key 77: 69677 -> 1877 -> 7977 -> NULL
Key 78: 13778 -> NULL
Key 79: 67979 -> NULL
Key 80: 82880 -> 38080 -> NULL
Key 81: 83481 -> NULL
Key 82: 95082 -> 682 -> NULL
Key 84: 46784 -> NULL
Key 85: 58185 -> 37585 -> NULL
Key 86: 24386 -> NULL
Key 94: 63294 -> 71994 -> NULL
Key 96: 60296 -> NULL
Key 97: 13297 -> NULL
Key 98: 62398 -> NULL
Collisions for table size 100: 38

```

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.

```

#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

```

Хеш-таблиця переповнена, неможливо вставити 3629
key: 0, value: 16271
key: 1, value: 72713
key: 2, value: 16522
key: 3, value: 82001
key: 4, value: 89259
key: 5, value: 93161
key: 6, value: 45688
key: 7, value: 72185
key: 8, value: 44955
key: 9, value: 70027
key: 0, value: 89259
key: 1, value: 16271
key: 2, value: 31665
key: 3, value: 72713
key: 4, value: 54591
key: 5, value: 15499
key: 6, value: 42072
key: 7, value: 45274
key: 8, value: 24693
key: 9, value: 59053
key: 10, value: 97427
key: 11, value: 57537
key: 12, value: 75535
key: 13, value: 51952
key: 14, value: 16522

```

Рисунок 1 - результат виконання програми