

Міністерство освіти і науки України
Національний університет "Львівська політехніка"
Інститут комп'ютерних наук та інформаційних технологій
Кафедра програмного забезпечення



Звіт

Про виконання лабораторної роботи №6
на тему:

«Особливості роботи з функціями в С.
Директиви препроцесора»
з дисципліни «Основи програмування»

Лектор:

ст. викл. каф. ПЗ
Муха Т.О.

Виконав:

ст. гр. ПЗ-11
Ясногородський Н.В.

Прийняв:

асист. каф. ПЗ
Дивак І.В.

« __ » _____ 2021 р.

Σ = _____ .

Львів – 2021

Тема: Особливості роботи з функціями в С. Директиви препроцесора.

Мета: Поглиблене вивчення можливостей функцій в мові С та основ роботи з препроцесором.

ЗАВДАННЯ

Завдання 1.

Написати програму для обробки даних, організованих у масив, згідно завдання наведеного варіанту.

5. Написати функцію для додавання двох матриць. З її допомогою додати вихідну матрицю і транспоновану.

Завдання 2.

5. У функцію зі змінним числом параметрів надходять додатні цілі числа, кінець списку – значення -1. Порахувати, скільки разів зустрічається кожна цифра у заданому числі.

Завдання 3.

5. Задати макрос обчислення n-ого члена геометричної прогресії за введеними користувачем першим членом і знаменником.

ТЕКСТ ПРОГРАМИ

Завдання 1

```
#include <stdio.h>
#include <math.h>
#define _USE_MATH_DEFINES

typedef struct Matrix
{
    double **values;
    int x;
    int y;
} matrix;

void initMatrix(matrix *matrix, int fillFromStd)
{
    matrix->values = (double **)malloc(matrix->y * sizeof(double *));

    for (int i = 0; i < matrix->y; i++)
    {
        matrix->values[i] = (double *)calloc(matrix->x, sizeof(double));
        if (fillFromStd)
        {
```

```

        printf("Enter %d elements for %d row:\n", matrix->x, i);
        for (int j = 0; j < matrix->x; j++)
        {
            printf("\tid: ", j);
            scanf("%lf", &matrix->values[i][j]);
        }
        printf("\n");
    }
}

```

```

void freeMatrix(matrix *matrix)
{
    for (int i = 0; i < matrix->x; i++)
    {
        free(matrix->values[i]);
    }
    free(matrix->values);
}

```

```

void addMatrix(matrix *a, matrix *b, matrix *result)
{
    if (a->x != b->x || a->y != b->y)
    {
        printf("Error: matrix sizes aren't equal!");
        return;
    }
}

```

```

result->x = a->x;
result->y = a->y;
initMatrix(result, 0);

```

```

for (int i = 0; i < a->y; i++)
{
    for (int j = 0; j < a->x; j++)
    {
        result->values[i][j] = a->values[i][j] + b->values[i][j];
    }
}
}

```

```

void printMatrix(matrix *matrix, char *name)
{
    printf("\n%s\n" matrix, name);
    for (int i = 0; i < matrix->y; i++)
    {
        for (int j = 0; j < matrix->x; j++)

```

```

    {
        printf("\t%.1lf", matrix->values[i][j]);
    }
    printf("\n");
}
printf("\n");
}

```

```

void transposeSquareMatrix(matrix *initial, matrix *out)

```

```

{
    if (initial->x != initial->y)
        return;

    out->x = initial->x;
    out->y = initial->y;
    initMatrix(out, 0);

    for (int i = 0; i < initial->y; i++)
        for (int j = 0; j < initial->x; j++)
            out->values[j][i] = initial->values[i][j];
}

```

```

int main(void)

```

```

{
    printf("Task 5, Section 1\n");
    printf("Enter matrix height and width: ");

    matrix a, b, result;
    int size = 0;
    scanf("%d", &size);
    a.x = a.y = size;

    initMatrix(&a, 1);
    printMatrix(&a, "Initial");

    transposeSquareMatrix(&a, &b);
    printMatrix(&b, "Transposed");

    addMatrix(&a, &b, &result);
    printMatrix(&result, "Sum");

    freeMatrix(&result);
    freeMatrix(&a);
    freeMatrix(&b);

    return 0;
}

```

Завдання 2

```
#include <stdio.h>
#include <stdarg.h>

void parseNumber(int);

void calcArgsDigits(int firstNumber, ...)
{
    int currentNum;
    va_list args;
    va_start(args, &firstNumber);

    parseNumber(firstNumber);
    // read args while current element not -1
    while ((currentNum = va_arg(args, int)) != -1)
    {
        parseNumber(currentNum);
    }
    va_end(args);
}

typedef struct DigitFrequency
{
    int digit;
    int count;
} digitFrequency;

void parseNumber(int number)
{
    int numberCopy = number;
    digitFrequency map[10];

    for (int i = 0; i < 10; i++)
    {
        map[i].digit = i;
        map[i].count = 0;
    }

    while (numberCopy)
    {
        int digit = numberCopy % 10;
        map[digit].count++;
        numberCopy /= 10;
    }

    printf("Parsing number \"%d\":\n", number);
    for (int i = 0; i < 10; i++)
    {
        printf("\t\"%d\" used %d times\n", map[i].digit, map[i].count);
    }
}

int main(void)
{
    printf("Task 5, Section 2\n");

    calcArgsDigits(1000, 22394, 3393939, 4999, 51111, 6120390756, -1);

    return 0;
}
```

Завдання 3

```
#include <stdio.h>

#include <math.h>

#define GET_N_GEOM_PROGRESSION_ELEMENT(firstEl, base, n) firstEl *pow(base,
n - 1)

int main(void)
{
    printf("Task 5, Section 3\n");

    double base, firstElement;

    printf("Enter base and first element: ");

    scanf("%lf %lf", &base, &firstElement);

    printf("Result: %.3lf\n", GET_N_GEOM_PROGRESSION_ELEMENT(firstElement,
base, 3));

    return 0;
}
```

РЕЗУЛЬТАТИ

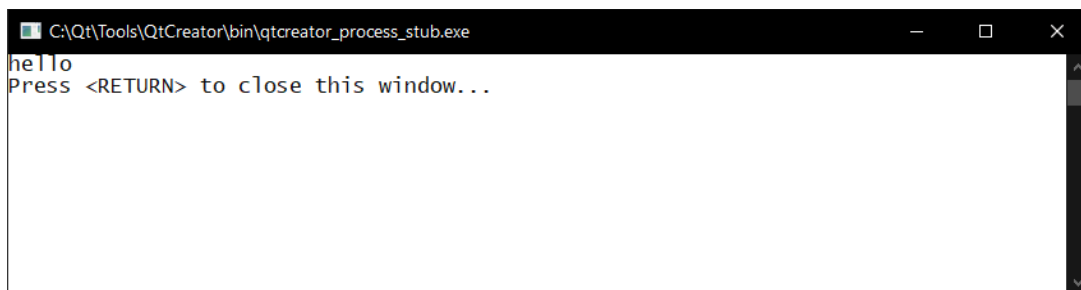
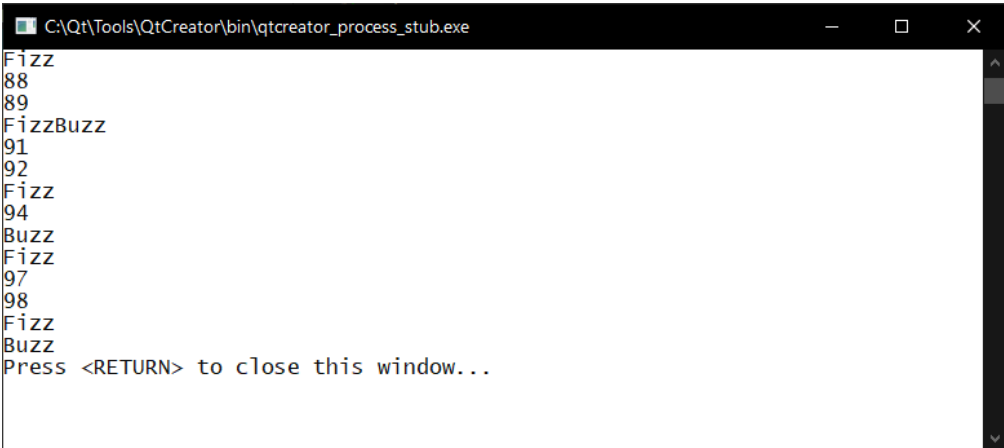


Рис 1. Результат виконання програми №1



```
C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe
Fizz
88
89
FizzBuzz
91
92
Fizz
94
Buzz
Fizz
97
98
Fizz
Buzz
Press <RETURN> to close this window...
```

Рис 2. Результат виконання програми №2

ВИСНОВКИ

Здобуто практичні навички створення та застосування функцій та макросів у мові C.