MEF University

Laboratory №6 - Report

Bogdan Itsam Dorantes-Nikolaev

Department of Engineering, MEF University

COMP 205-02: Systems Programming

Prof. Buse Yılmaz & Assistant Ayşenaz Ezgi Ergin

November 25, 2022

1. Problem

This program provided a text file, will output a CSR formatted sparse matrix over the console. The inputted text file, which comprises the number of rows and columns and float values separated by commas, is converted into the console output using three dynamically constructed arrays labeled vals, col_idx, and row_ptr.

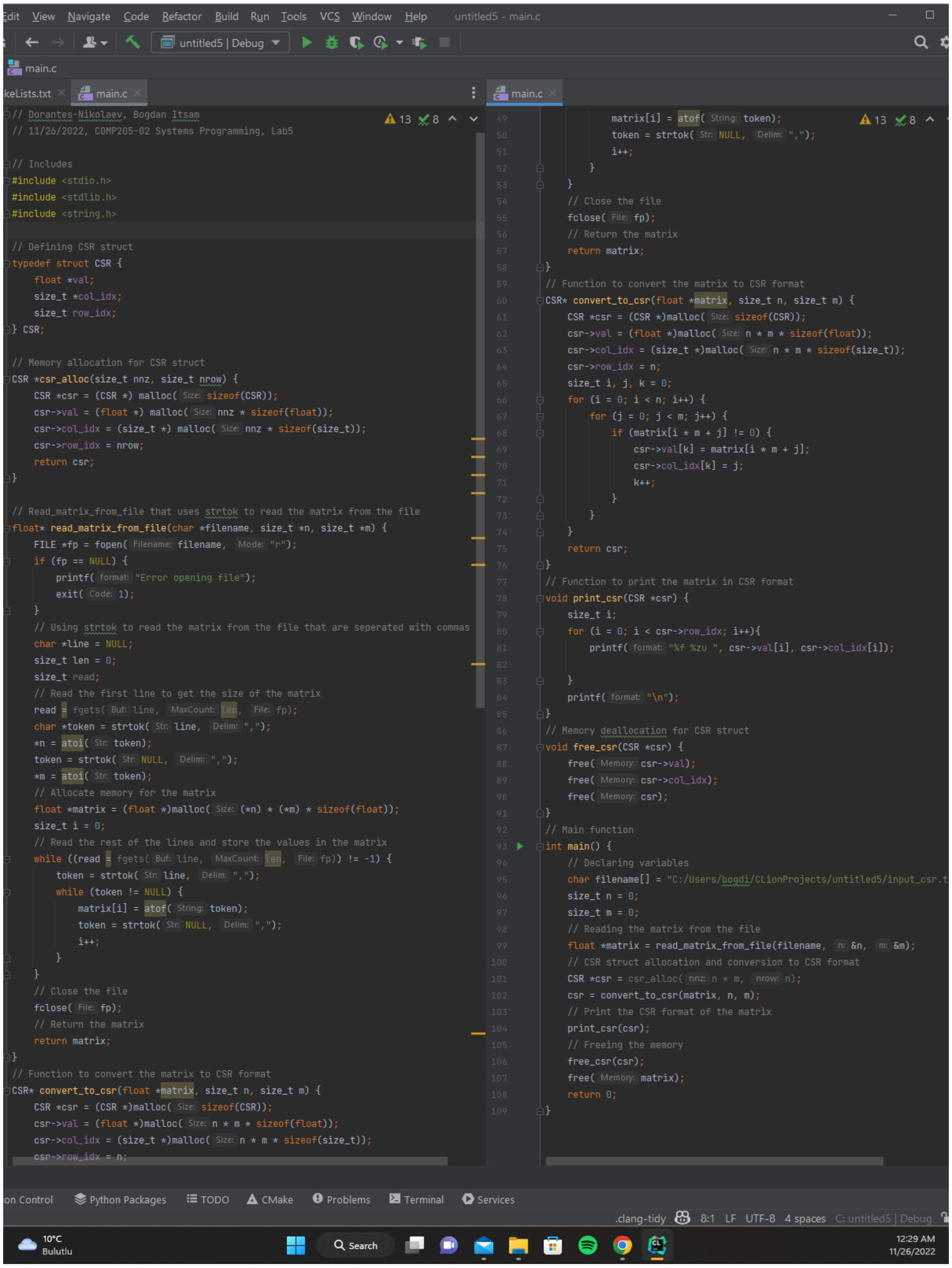
2. Solution (Pseudocode)

```
#include studio.h & stdlib.h & string.h

struct CSR
    Float *val
    Size_t *col_idx & row_idx
read_matrix_from_file(char *filename, size_t *n, size_t *m)
    Open file, if file empty return -1 + error
    Initialize variables to read file
    Read using fgets and get matrix size
    Split string using strtok & tokens
    Allocate memory for matrices
    Split each element and record to new matrix
    Close input and return matrix
convert_to_csr(float *matrix, size_t n, size_t m)
    Allocate memory for *csr, val, col_idx,row_idx
    Nested for loops to iterate through rows & columns
       return csr
print_csr(CSR *csr)size_t i
    Initialize for loop variables
    Nested for loop iterating through column and row
    Print newline
free_csr(CSR *csr)
    Free csr->val & csr->col_idx & csr
Main()
    Filename decleration
    Matrix size initialization
    Reading the matrix from the file
    Allocating CSR memory
    Converting the matrix to CSR format
    Print the csr format of the matrix
    Freeing the memory
```

Table.1: Pseudocode of the text file to CSR program

3. Code - 3.1 - Matrix Fill:



```c
// Dorantes-Nikolaev, Bogdan Itsam
// 11/26/2022, COMP205-02 Systems Programming, Lab5

// Includes
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Defining CSR struct
typedef struct CSR {
    float *val;
    size_t *col_idx;
    size_t row_idx;
} CSR;

// Memory allocation for CSR struct
CSR *csr_alloc(size_t nnz, size_t nrow) {
    CSR *csr = (CSR *) malloc( sizeof(CSR));
    csr->val = (float *) malloc( nnz * sizeof(float));
    csr->col_idx = (size_t *) malloc( nnz * sizeof(size_t));
    csr->row_idx = nrow;
    return csr;
}

// Read_matrix_from_file that uses strtok to read the matrix from the file
float* read_matrix_from_file(char *filename, size_t *n, size_t *m) {
    FILE *fp = fopen( filename, "r");
    if (fp == NULL) {
        printf( "Error opening file");
        exit( 1);
    }
    // Using strtok to read the matrix from the file that are seperated with commas
    char *line = NULL;
    size_t len = 0;
    size_t read;
    // Read the first line to get the size of the matrix
    read = fgets( line, len, fp);
    char *token = strtok( line, ",");
    *n = atoi( token);
    token = strtok( NULL, ",");
    *m = atoi( token);
    // Allocate memory for the matrix
    float *matrix = (float *)malloc( (*n) * (*m) * sizeof(float));
    size_t i = 0;
    // Read the rest of the lines and store the values in the matrix
    while ((read = fgets( line, len, fp)) != -1) {
        token = strtok( line, ",");
        while (token != NULL) {
            matrix[i] = atof( token);
            token = strtok( NULL, ",");
            i++;
        }
    }
    // Close the file
    fclose( fp);
    // Return the matrix
    return matrix;
}
// Function to convert the matrix to CSR format
CSR* convert_to_csr(float *matrix, size_t n, size_t m) {
    CSR *csr = (CSR *)malloc( sizeof(CSR));
    csr->val = (float *)malloc( n * m * sizeof(float));
    csr->col_idx = (size_t *)malloc( n * m * sizeof(size_t));
    csr->row_idx = n;
```

```c
            matrix[i] = atof( token);
            token = strtok( NULL, ",");
            i++;
        }
    }
    // Close the file
    fclose( fp);
    // Return the matrix
    return matrix;
}
// Function to convert the matrix to CSR format
CSR* convert_to_csr(float *matrix, size_t n, size_t m) {
    CSR *csr = (CSR *)malloc( sizeof(CSR));
    csr->val = (float *)malloc( n * m * sizeof(float));
    csr->col_idx = (size_t *)malloc( n * m * sizeof(size_t));
    csr->row_idx = n;
```

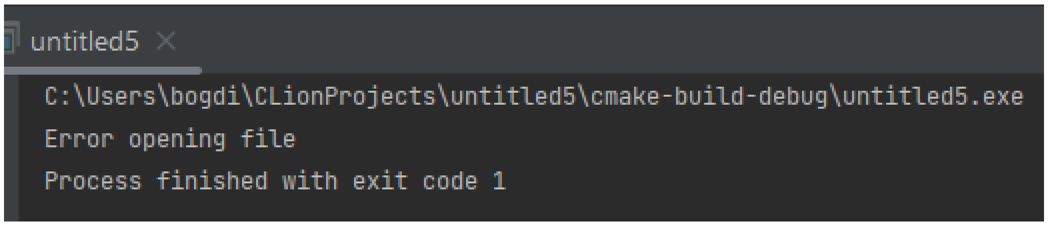
```c
    size_t i, j, k = 0;
    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
            if (matrix[i * m + j] != 0) {
                csr->val[k] = matrix[i * m + j];
                csr->col_idx[k] = j;
                k++;
            }
        }
    }
    return csr;
}
// Function to print the matrix in CSR format
void print_csr(CSR *csr) {
    size_t i;
    for (i = 0; i < csr->row_idx; i++){
        printf( "%f %zu ", csr->val[i], csr->col_idx[i]);

    }
    printf( "\n");
}
// Memory deallocation for CSR struct
void free_csr(CSR *csr) {
    free( csr->val);
    free( csr->col_idx);
    free( csr);
}
// Main function
int main() {
    // Declaring variables
    char filename[] = "C:/Users/bogdi/CLionProjects/untitled5/input_csr.t
    size_t n = 0;
    size_t m = 0;
    // Reading the matrix from the file
    float *matrix = read_matrix_from_file(filename, &n, &m);
    // CSR struct allocation and conversion to CSR format
    CSR *csr = csr_alloc( n * m, n);
    csr = convert_to_csr(matrix, n, m);
    // Print the CSR format of the matrix
    print_csr(csr);
    // Freeing the memory
    free_csr(csr);
    free( matrix);
    return 0;
}
```

Figure.1 IDE Code screenshot of Matrix Fill program

4. Console Outputs

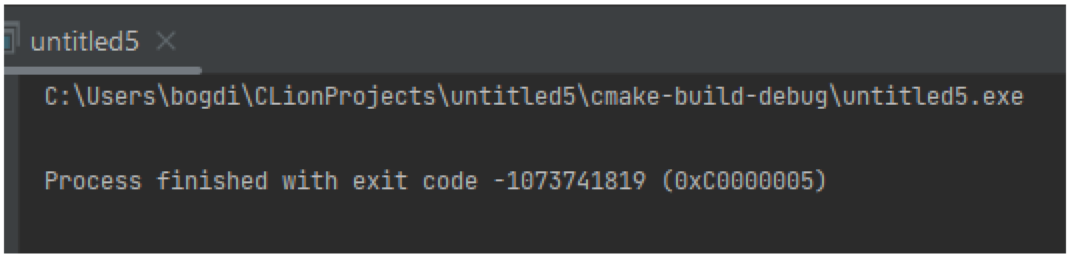
Case 1: File Read Error:



```
C:\Users\bogdi\CLionProjects\untitled5\cmake-build-debug\untitled5.exe
Error opening file
Process finished with exit code 1
```

Table.2 Output of Matrix Fill Program when an invalid input was given

Case 2: Memory Error:

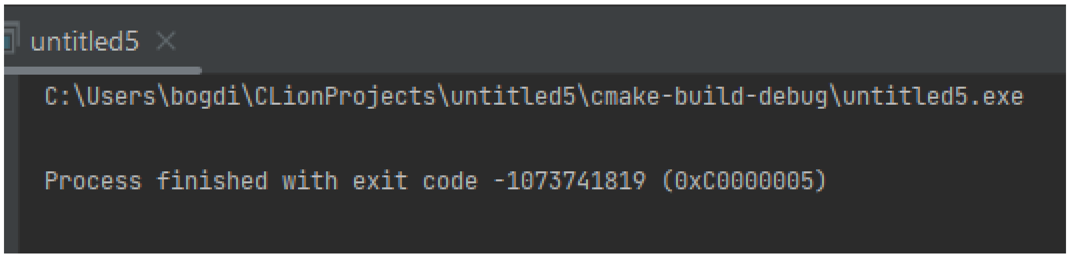


```
C:\Users\bogdi\CLionProjects\untitled5\cmake-build-debug\untitled5.exe

Process finished with exit code -1073741819 (0xC0000005)
```

Table.3 Output of Matrix Fill Program when memory error occurs

Case 3: CSR



```
C:\Users\bogdi\CLionProjects\untitled5\cmake-build-debug\untitled5.exe

Process finished with exit code -1073741819 (0xC0000005)
```

Table.3 Attempted output for CSR