```cpp
// matrices_txt_stream.cpp
//
#ifdef _WIN32
    #pragma warning(disable:4996)
    #include <tchar.h>
    #include <windows.h>
    #include <conio.h>
    #include <direct.h>
#elif (defined __linux__) || (defined _AIX) || (defined __APPLE__)
    #include <stdlib.h>
    #include <sys/types.h>
    #include <sys/stat.h>
    #include <unistd.h>
    typedef char _TCHAR;
    #define _tmain main
#endif

#include <string.h>
#include <fstream>
#include <iostream>
#include <iomanip>
using namespace std;

#define MAXLINE 255
#define MAXLEN 20
typedef float MATTYPE[MAXLEN][MAXLEN];

void my_getch();
void GetNamesOfIOFiles(const char[], char[], const char[], char[], char[]);

int _tmain(int argc, _TCHAR* argv[])
{
    ifstream in;
    ofstream out;
    FILE* fileC;
    MATTYPE A, B, C;
    float Suma;
    int m, n, r, s, i, j;
    char name[9], input_file[MAXLINE], output_file[MAXLINE],
        pure_output_path[MAXLINE];

    cout << "\nTento program nacita z textoveho suboru matice A a B\n";
    cout << "a maticu C=A*B vypise na obrazovku a do textoveho suboru.\n\n";

    GetNamesOfIOFiles("MATICE.TXT", input_file, "VYSLEDOK.TXT", output_file,
        pure_output_path);
    in.open(input_file, ios::in);
    in >> m >> n;
    if (in.fail()) {
        cout << "Subor MATICE.TXT sa nepodarilo otvorit.\n";
        my_getch();
        return 1;
    }
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            in >> A[i][j];
    in >> r >> s;
    if (n != r) {
        cout << "Matice sa nedaju nasobit!";
        my_getch();
        return 1;
    }
    for (i = 0; i < r; i++)
        for (j = 0; j < s; j++)
            in >> B[i][j];
    in.close();

    for (i = 0; i < m; i++)
        for (j = 0; j < s; j++) {
            Suma = 0;
            for (int k = 0; k < n; k++)
                Suma += A[i][k] * B[k][j];
            C[i][j] = Suma;
        }

    cout << "Z textoveho suboru sa nacitali 2 matice.\n\nA:";
    cout.setf(ios::fixed, ios::floatfield);
    cout.precision(2);
    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++)
            cout << setw(8) << A[i][j];
        cout << "\n  ";
    }
    cout << "\nB:";
    for (i = 0; i < r; i++) {
        for (j = 0; j < s; j++)
            cout << setw(8) << B[i][j];
        cout << "\n  ";
    }
    cout << "\nVysledna matica C=A*B je:\n\n";
    for (i = 0; i < m; i++) {
        cout << "  ";
        for (j = 0; j < s; j++)
            cout << setw(8) << C[i][j];
        cout << "\n";
    }

    if ((fileC = fopen(output_file, "r")) != NULL) {
        fclose(fileC);
        cout << "\nVystupny subor VYSLEDOK.TXT uz existuje.\n\n";
        cout << "  1. Chcete prepisat tento subor?\n";
        cout << "  2. Chcete pripisat nove vysledky na koniec tohto suboru?\n";
        cout << "  3. Chcete zadat novy nazov pre vystupny subor?\n\n";
        do {
            cout << "Volba=";
            cin >> i;
        } while ((i < 1) || (i > 3));
        switch (i) {
        case 1: out.open(output_file, ios::out);
            break;
        case 2: out.open(output_file, ios::app);
            break;
        case 3: cout << "\nNapiste novy nazov suboru s maximalne 8 znakmi a ";
            cout << "bez pripony: ";
            scanf("%8s", name);
            strcat(name, ".txt");
            strcpy(output_file, pure_output_path);
            strcat(output_file, name);
            out.open(output_file, ios::out);
        }
    }
    else
        out.open(output_file, ios::out);

    out << "Z textoveho suboru sa nacitali 2 matice.\n";
    out.setf(ios::fixed, ios::floatfield);
    out.precision(2);
    out << "\nA:";
    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++)
            out << setw(8) << A[i][j];
        out << "\n  ";
    }
    out << "\nB:";
    for (i = 0; i < r; i++) {
        for (j = 0; j < s; j++)
            out << setw(8) << B[i][j];
        out << "\n  ";
    }
    out << "\nVysledna matica C=A*B je:\n";
    out << "\nC:";
    for (i = 0; i < m; i++) {
        for (j = 0; j < s; j++)
            out << setw(8) << C[i][j];
        out << "\n  ";
    }
```

```
    out << "\n";
    out.close();

    my_getch();
    return 0;
}
//---------------------------------------------------------------------------
void my_getch()
{
#ifdef _WIN32
    _getch();
#else
    cout << endl;
#endif
}
//---------------------------------------------------------------------------
void GetNamesOfIOFiles(const char name_of_input_file[], char path_to_input_file[],
    const char name_of_output_file[], char path_to_output_file[],
    char pure_output_path[])
{
    char current_path[MAXLINE];
    current_path[0] = '\0';

#ifdef _WIN32
    TCHAR exePath[MAXLINE];

    HMODULE hModule = GetModuleHandle(NULL);
    if (hModule != NULL) {
        if (!GetModuleFileName(hModule, exePath, MAXLINE)) {
            cout << "Nepodarila sa zistit cesta k exe-suboru.\n";
            my_getch();
            exit(1);
        }
    }
    else {
        cout << "Module handle is NULL.\n" << endl;
        my_getch();
        exit(1);
    }

    int iii;
    bool flag = false;
    for (iii = (int)wcslen(exePath); iii >= 0; iii--) {
        if (!flag && exePath[iii] == '\\') {
            current_path[iii + 1] = '\0';
            flag = true;
        }
        if (flag)
            current_path[iii] = (char)exePath[iii];
    }
#elif (defined __linux__) || (defined __APPLE__)
    unsigned iii;
    char line[MAXLINE];
    FILE* fp;
    if ((fp = popen("/bin/pwd", "r")) == NULL) {
        perror("popen error");
        exit(1);
    }
    if (fgets(line, MAXLINE, fp) == NULL) {
        perror("fgets error");
        exit(1);
    }
    pclose(fp);

    iii = 0;
    while (line[iii] != '\r' && line[iii] != '\n') {
        current_path[iii] = line[iii];
        iii++;
    }
    current_path[iii] = '\0';
#elif (defined _AIX)
    unsigned iii;
    char line[MAXLINE];
```

```c
    FILE* fp;
    if ((fp = popen("user/bin/pwd", "r")) == NULL) {
        perror("popen error");
        exit(1);
    }
    if (fgets(line, MAXLINE, fp) == NULL) {
        perror("fgets error");
        exit(1);
    }
    pclose(fp);

    iii = 0;
    while (line[iii] != '\r' && line[iii] != '\n') {
        current_path[iii] = line[iii];
        iii++;
    }
    current_path[iii] = '\0';
#endif

    path_to_input_file[0] = '\0';
    strcat(path_to_input_file, current_path);
#if (defined __linux__) || (defined _AIX) || (defined __APPLE__)
    strcat(path_to_input_file, "/inputs/");
#elif (defined _WIN32)
    strcat(path_to_input_file, "inputs\\");
#endif
    strcat(path_to_input_file, name_of_input_file);

    path_to_output_file[0] = '\0';
    strcat(path_to_output_file, current_path);
#if (defined __linux__) || (defined _AIX) || (defined __APPLE__)
    struct stat st = { 0 };
    strcat(path_to_output_file, "/outputs/");

    if (stat(path_to_output_file, &st) == -1)
        mkdir(path_to_output_file, 0755);
#elif (defined _WIN32)
    strcat(path_to_output_file, "outputs\\");
    if (_mkdir(path_to_output_file) != 0)
        if (errno == ENOENT) {
            perror("_mkdir error");
            exit(1);
        }
#endif
    pure_output_path[0] = '\0';
    strcpy(pure_output_path, path_to_output_file);
    strcat(path_to_output_file, name_of_output_file);
}
//----------------------------------------------------------------------------
```