```cpp
// exponentiate_matrix.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 <stdio.h>
#include <string.h>
#include <iostream>
using namespace std;

#define MAXLINE 255
#define MAXLEN 20

void my_getch();
void NacitajMaticu(char*, unsigned&, unsigned&, float[][20]);
void VynasobMatice(const unsigned, const unsigned, const unsigned,
    const float[][20], const float[][20], float[][20]);
void UmocniMaticu(const unsigned, const unsigned, const float[][20], float[][20]);
void ZapisMaticu(FILE*, const unsigned, const unsigned, const unsigned,
    const float[][20], const float[][20]);
void RovnaSa(const unsigned, const unsigned, const float[][20], float[][20]);
void GetNamesOfIOFiles(const char[], char[], const char[], char[]);

int _tmain(int argc, _TCHAR* argv[])
{
    float A[MAXLEN][MAXLEN], B[MAXLEN][MAXLEN];
    unsigned m, n, Exponent;
    char input_file[MAXLINE], output_file[MAXLINE];
    FILE* out;

    cout << "\n  Tento program nacita z textoveho suboru stvorcovu maticu A\n"
        << "a z klavesnice prirodzene cislo k a na obrazovku a do textoveho \n"
        << "suboru vypise k-tu mocninu matice A.";

    GetNamesOfIOFiles("MATICA.TXT", input_file, "VYSLEDOK.TXT", output_file);
    NacitajMaticu(input_file, m, n, A);
    cout << "\n\nZadajte prirodzene cislo ako exponent matice A!\nExponent=";
    cin >> Exponent;

    UmocniMaticu(Exponent, n, A, B);

    ZapisMaticu(stdout, m, n, Exponent, A, B);
    out = fopen(output_file, "w");
    ZapisMaticu(out, m, n, Exponent, A, B);
    fclose(out);

    my_getch();
    return 0;
}
//---------------------------------------------------------------------------
void my_getch()
{
#ifdef _WIN32
    _getch();
#else
    cout << endl;
#endif
}
//---------------------------------------------------------------------------
void NacitajMaticu(char* MenoSuboru, unsigned& m, unsigned& n, float X[][20])
{
    FILE* in;

    if ((in = fopen(MenoSuboru, "r")) == NULL) {
        cout << "\n\n  Subor MATICA.TXT sa nepodarilo otvorit.\n";
        my_getch();
        exit(1);
    }
    fscanf(in, "%u%u", &m, &n);
    for (unsigned i = 0; i < m; i++)
        for (unsigned j = 0; j < n; j++)
            fscanf(in, "%f", &X[i][j]);
    fclose(in);
}
//---------------------------------------------------------------------------
void VynasobMatice(const unsigned m, const unsigned n, const unsigned o,
    const float X[][20], const float Y[][20], float Z[][20])
{
    for (unsigned i = 0; i < m; i++)
        for (unsigned j = 0; j < o; j++) {
            float Suma = 0;
            for (unsigned k = 0; k < n; k++)
                Suma += X[i][k] * Y[k][j];
            Z[i][j] = Suma;
        }
}
//---------------------------------------------------------------------------
void UmocniMaticu(const unsigned k, const unsigned n, const float X[][20],
    float Y[][20])
{
    unsigned kk = k;
    float Pom[20][20];

    switch (kk) {
    case 0:
        for (unsigned i = 0; i < n; i++) {
            for (unsigned j = 0; j < n; j++)
                Y[i][j] = 0;
            Y[i][i] = 1;
        }
        break;
    case 1: RovnaSa(n, n, X, Y);
        break;
    default: RovnaSa(n, n, X, Pom);
        kk--;
        while (kk--) {
            VynasobMatice(n, n, n, X, Pom, Y);
            RovnaSa(n, n, Y, Pom);
        }
    }
}
//---------------------------------------------------------------------------
void ZapisMaticu(FILE* out, const unsigned m, const unsigned n, const unsigned k,
    const float X[][20], const float Y[][20])
{
    unsigned i, j;

    fprintf(out, "\nZ textoveho suboru sa nacitala matica A:\n");
    for (i = 0; i < m; i++) {
        fprintf(out, "\n  ");
        for (j = 0; j < n; j++)
            fprintf(out, "%10.2f", X[i][j]);
    }
    fprintf(out, "\n");
    fprintf(out, "\n%u-ta mocnina matice A je:\n", k);
    for (i = 0; i < m; i++) {
        fprintf(out, "\n  ");
        for (j = 0; j < n; j++)
            fprintf(out, "%10.2f", Y[i][j]);
    }
    fprintf(out, "\n");
}
//---------------------------------------------------------------------------
void RovnaSa(const unsigned m, const unsigned n, const float A[][20], float B[][20])
```

```cpp
{
    for (unsigned i = 0; i < m; i++)
        for (unsigned j = 0; j < n; j++)
            B[i][j] = A[i][j];
}
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
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 current_path[MAXLINE];

#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];
    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
    strcat(path_to_output_file, name_of_output_file);
}
//----------------------------------------------------------------------------
```