```cpp
// W17_0.cpp : Defines the entry point for the console application.
//             Demonstruje:
//             1. instrukcji bloku try moga byc bardzo glebokiego wlozenia
//             2. po wygenerowaniu exception sterowanie jest przekazywane bloku catch
//                z wyjatkem odpowiedniego typu
//             3. pozostala czesc kodu od momentu wygenerowania exception i do
//                konca bloka try nigdy nie bedzie wykonana
//             4. argument arg instrukcji catch(typ arg) jest automatycznie
//                ustawiony na ta wartosc wyjatku danego typu, ktora pozostala
//                okreslona przy uzyciu throw arg
//             5. jesli pozostaje wygenerowany wyjatek, typ ktorego nie jest oznaczony
//                w instrukcji catch, program bedzie przerwany
//             6. Zeby zadzialal my_terminate, uruchom program bezposrednio z katalogu
//                debug lub release. Inaczej debuger VS podpina swoj terminate handler

#include "stdafx.h"
#include <iostream>
//#include <exception>
using namespace std;

int my_except[] = {100, 200};

void fun(int ii);
void my_terminate();

int _tmain(int argc, _TCHAR* argv[])
{
    int i = 0;

    try
    {
        //podpinamy swoj terminate handler
        terminate_handler old_term = set_terminate(my_terminate);

        cout << "Input i = 0, 1\n";
        cin >> i;
        cout << "start from try\n";
        fun(i);
        cout << "this part of code newer been produced\n";
    }
    catch(int excp)
    {
        switch(excp)
        {
        case 100:
            cout << "exception my_except 100\n";
            break;
        case 200:
            cout << "exception my_except 200\n";
            break;
        }
    }

    system("pause");
    return 0;
}

void fun(int ii)
{
    if(ii == 0)
        throw my_except[0];
    else if(ii == 1)
        throw my_except[1];
    else
        throw 'a';    //a jest typem char, a nie int
}

void my_terminate()
{
    cout << "to ja, my_terminate" << endl;
    system("pause");
    exit(-1);
}
```