KALKULATOR MACIERZY

Autor: Michał Rafałowski

Akademia Górniczo-Hutnicza im. Stanisława Staszica

w Krakowie

Kraków 2018

Spis treści

1. Wstęp

Dokumentacja projektowa dotyczy wstępnego opracowania i analizy problemu kalkulatora, którego celem będzie wykonywanie operacji na macierzach. Ponadto do dyspozycji użytkownika będą dostępne opisy, jak działają wszystkie funkcje dostępne w programie.

Na podstawie wstępnej dokumentacji został napisany kod opisany szerzej w punkcie 6. Opis realizacji.

Opracowany kalkulator macierzy jest projektem zaliczeniowym z przedmiotu „Języki programowania obiektowego" na kierunku „elektronika i telekomunikacja" Wydziału Informatyki, Elektroniki i Telekomunikacji Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie.

2. Wymagania systemowe

Podstawowe założenia projektu:

1. Określenie funkcji kalkulatora, zrozumienie ich oraz odpowiednie zaimplementowanie.
2. Określenie w jaki sposób macierz będzie reprezentowana.
3. Przygotowanie interfejsu kalkulatora używając wiersza poleceń.
4. Określenie sposobu w jaki zostaną zaprezentowane na ekranie wyniki działań.
5. Określenie w jaki sposób użytkownik będzie wprowadzał dane.

3. Funkcjonalność

Zostanie przygotowane menu. Każda funkcja kalkulatora będzie miała przypisane unikatowe oznaczenie w postaci numeru. Użytkownik będzie dokonywał wyboru poprzez wpisanie liczby symbolizującej odpowiednią opcję w menu.

Dostępne funkcje w menu:

Dla jednej macierzy:

-mnożenie przez liczbę
-wyznacznik (dla macierzy 2x2 oraz 3x3)
-transpozycja

Dla dwóch macierzy:

-dodawanie
-odejmowanie
-mnożenie

Opisy funkcji:

-mnożenie przez liczbę
-wyznacznik (dla macierzy 2x2 oraz 3x3)
-transpozycja
-dodawanie
-odejmowanie
-mnożenie

Wyjście z aplikacji.

4. Analiza problemu

Macierz reprezentowana będzie poprzez wektor wektorów elementów zmiennoprzecinkowych typu double.

Należy przeanalizować każdą z dostępnych operacji na macierzach.

Przyjęte oznaczenia:

a_{ij} - element macierzy A o współczynniku ij, gdzie i oznacza wiersz, a j kolumnę

b_{ij} - element macierzy B o współczynniku ij, gdzie i oznacza wiersz, a j kolumnę

c_{ij} - element macierzy C, będącej wynikiem działań na macierzach A i B, o współczynniku ij, gdzie i oznacza wiersz, a j kolumnę

x – liczba zmiennoprzecinkowa

4.1 Dodawanie macierzy

Dodawanie macierzy jest dwuargumentowym działaniem, polegającym na sumowaniu elementów obu macierzy o tych samych parametrach wiersz/kolumna. Warunkiem koniecznym tej operacji jest to, aby wymiary oby macierzy były identyczne.

$$c_{ij} = a_{ij} + b_{ij}. \quad (1)$$

4.2 Odejmowanie macierzy

Odejmowanie macierzy jest dwuargumentowym działaniem, polegającym na obliczeniu różnicy elementów obu macierzy o tych samych parametrach wiersz/kolumna. Warunkiem koniecznym tej operacji jest to, aby wymiary macierzy były identyczne.

$$c_{ij} = a_{ij} - b_{ij}. \quad (2)$$

4.3 Mnożenie dwóch macierzy

Mnożenie dwóch macierzy jest dwuargumentowym działaniem, które realizowane jest poprzez zastosowanie operacji nazwanej „mnożeniem Cauchy'ego" do wyznaczania każdego elementu macierzy końcowej. Warunkiem koniecznym tej operacji jest, to aby pierwsza macierz miała tyle kolumn, ile druga wierszy .

$$c_{i,j} = \sum_{r=1}^{m} a_{i,r} b_{r,j} = a_{i,1} b_{1,j} + a_{i,2} b_{2,j} + \cdots + a_{i,m} b_{m,j} \quad (3)$$

Oznaczenia do wzoru (3) [1]:

A - macierz o wymiarach n x m, gdzie n oznacza wiersz, a m kolumnę

B - macierz o wymiarach m x p, gdzie m oznacza wiersz, a p kolumnę

C - macierz o wymiarach n x p, gdzie n oznacza wiersz, a p kolumnę

a_{ir} - element macierzy A o współczynnikach ij, gdzie i oznacza wiersz, a j kolumnę

b_{rj} - element macierzy B o współczynnikach rj, gdzie r oznacza wiersz, a j kolumnę

c_{ij} - element macierzy C, będącej wynikiem działań na macierzach A i B, o współczynnika ij, gdzie i oznacza wiersz, a j kolumnę

r – zmienna typu całkowitego o początkowej wartości 1 i końcowej m

4.4 Mnożenie macierzy przez liczbę

Mnożenie macierzy przez liczbę jest dwuragumetowym działaniem, polegającym na przemnożeniu każdego elementu danej macierzy przez podany skalar.

$$c_{ij} = a_{ij} * x \tag{4}$$

4.5 Wyznacznik macierzy

Wyznacznik – funkcja przyporządkowująca każdej macierzy kwadratowej $M_{n,n}(R)$o współczynnikach z pierścienia przemiennego R pewien element tego pierścienia[2].
W przypadku omawianego kalkulatora pod uwagę będziee brane obliczanie wyznacznika macierzy 2x2 bądź 3x3 poprzez odpowiednio metodę „na krzyż” (5)[3] oraz metodę Sarrusa” (6)[4].

$$\det\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = a_{11}a_{22} - a_{12}a_{21} \tag{5}$$

$$det\begin{bmatrix} 0 & 3 & 1 \\ 1 & -1 & 2 \\ 2 & 4 & 1 \end{bmatrix}\begin{matrix} 0 & 3 \\ 1 & -1 \\ 2 & 4 \end{matrix} = \tag{6}$$

$$= (0\cdot(-1)\cdot 1+3\cdot 2\cdot 2+1\cdot 1\cdot 4)-(2\cdot(-1)\cdot 1+4\cdot 2\cdot 0+1\cdot 1\cdot 3)$$

4.6 Transpozycja

Transponowanie macierzy polega na zamianie wierszy macierzy na kolumny oraz kolumny na wiersze.

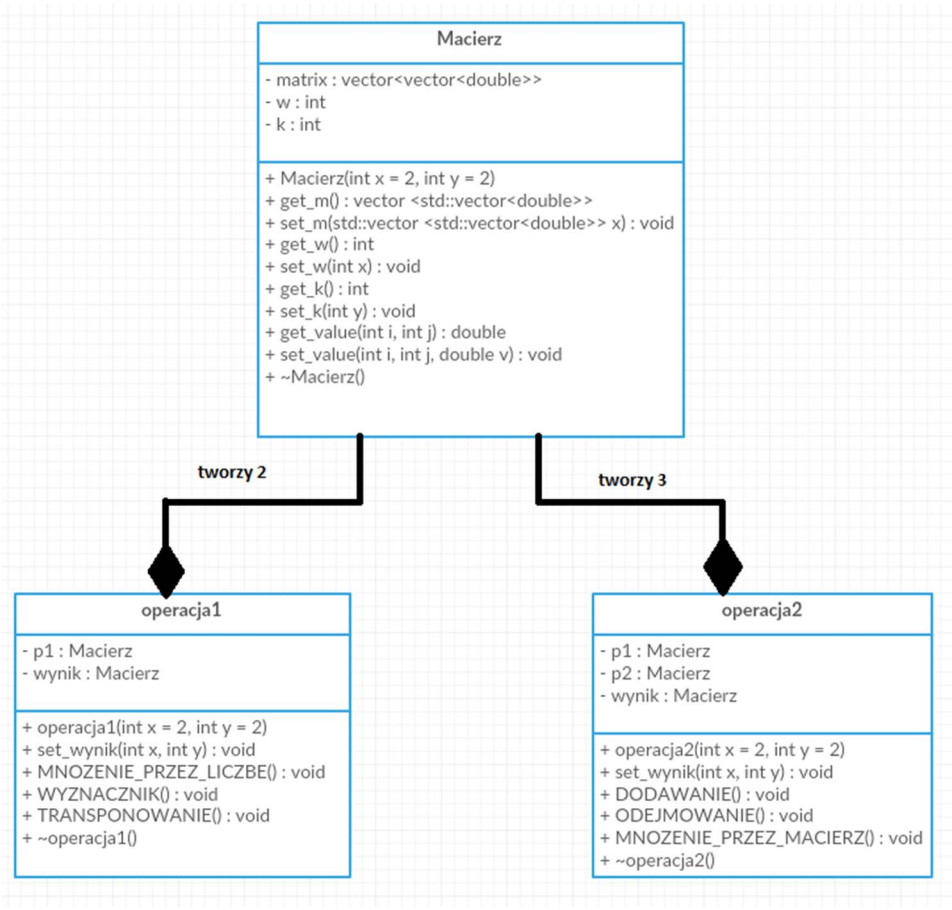
$$c_{ij} = a_{ji} \tag{7}$$

4.7 Operacje wejścia/wyjścia:

Operacje będą realizowane poprzez 2 pętle typu for sterujące współczynnikami macierzy. Wewnątrz pętli będą dokonywać się odpowiednie operacje. Zastosowanie tych pętli będzie też konieczne podczas wpisywania oraz wypisywania elementów macierzy.

5. Projekt techniczny

W kodzie znajdują się 3 klasy: „Macierz”, „operacja1” oraz „operacja2”. Klasa „Macierz” zawiera w sobie atrybuty do reprezntowania macierzy oraz funkcjie „get/set” do operowania atrybutami. Klasy „operacja1” oraz „operacja2” zajmują się reprezentowaniem operacji na odpowednio jednej oraz dwóch macierzach. Występuje zależność pomiędzy tymi klasami a klasą „Macierz”. Ich atrybutami są obiekty klasy „Macierz” przechowujące podawane przez użytkownika macierze oraz macierz bedącą wynikiem danej operacji. Metodami tych klas są odpowiednie działania przeznaczone dla jednej lub drugiej macierzy.



Rys. 5-1 Diagram UML.

6. Opis realizacji

6.1 Opis pisania kodu

Kod został napisany na prywatnym komputerze z systemem Windows 7 64bit. Został użyty kompilator Visual Studio Enterprise 2017 w wersji 4.7.02558.

Aktualizacja 02.2018: Kalkulator obsługuje liczby zmiennoprzecinkowe. Zostały też wprowadzone klasy operacja1 i operacja2 oraz podział na kilka plików w kodzie.

Pisanie kodu zacząłem od zaimplementowania klasy „Macierz" zajmującej się reprezentacją macierzy. Jest ona najbardziej rozbudownym fragmentem kodu, kluczowym do działania całego programu. Posiada ona 3 atrybuty:

-zmienną typu int o nazwie „w" przechowującej liczbę wierszy
-zmienną typu int o nazwie „k**"** przechowującej liczbę kolumn
- vector vectorów typu int o nazwie „matrix" przechowujący właściwą macierz (Był to najlepszy wybór ze wszystkich, ponieważ zwykła tablica dwuwymiarowa wymaga z góry określonego rozmiaru, a jej dynamiczna wersja ma niepotrzebnie przekomplikowany sposób implementacji)

Następnie zostały zaimplementowane podstawowe metody klasy takie jak:

1. Konstruktor defaultowy oraz parametryczny, tworzący domyślnie macierz 2x2 i wypełniający ją wartościami z przedziału 0-9
2. Funkcja wypisująca elementy macierzy w zależności od ilości symbolów elementu.
3. Funkcje get/set atrybutów oraz pojedynczych elementow macierzy
4. Destuktor

Nastepnym krokiem były funkcje do podawania przez użytkownika wymiarów i elementów macierzy „set_elem", „set_row" oraz „set_column", które w połączeniu z setterami klasy Macierz ustawiają jej istotne parametry.

Klasa „operacja1" realizuje funkcje kalkulatora dla jednej macierzy. Jej atrybutami są macierz „p1", którą użytkownik podaje do obliczenia, oraz macierz „wynik" przechowująca późniejszy wynik.

Metody tej klasy to:

1. Mnożenie macierzy przez podaną liczbę
2. Obliczanie wyznacznika podanej macierzy 2x2 lub 3x3
3. Transponowanie macierzy

Klasa „operacja2” realizuje funkcje kalkulatora dla dwóch macierzy. Jej atrybutami są macierze „p1” oraz „p2”, które użytkownik podaje do obliczenia, oraz macierz „wynik” przechowująca późniejszy wynik.

Metody tej klasy to:

1. Dodawanie dwóch macierzy
2. Odejmowania dwóch macierzy
3. Mnożenie dwóch macierzy przez siebie

Każda funkcja, która tego wymagała, została uzupełniona o pętle „do...while” sprawdzającą, czy warunki danej operacji zostały spełnione.

Funkcje wypisujące opisy poszczególnych operacji zostały przeniesione z klas „operacja1” i „operacja2” do oddzielnego pliku źródłowego „Tutorial.cpp”.

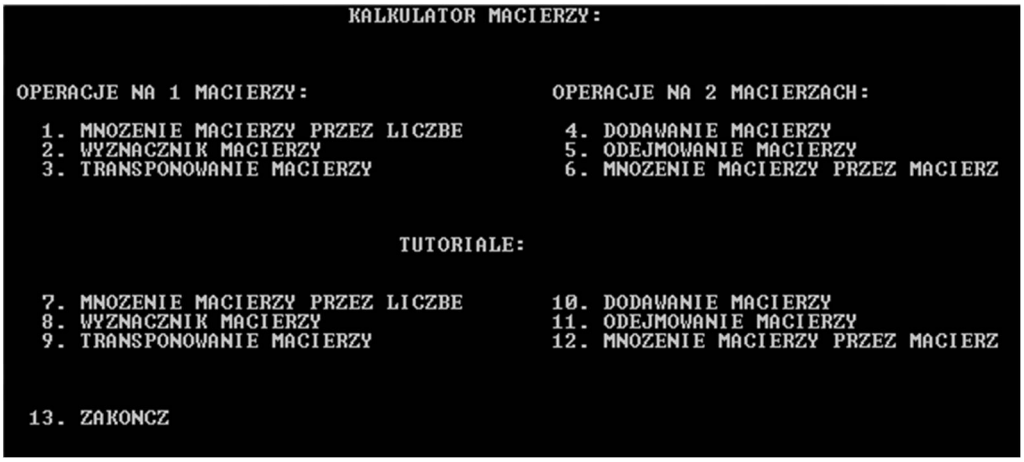
Ostatnim krokiem było przygotowanie menu, w którym użytkownik łatwo mógłby się odnaleźć. Dlatego też powstała funkcja „menu()” zajmująca się wyświetlaniem na ekranie wiersza poleceń wszystkich dostępnych funkcji kalkulatora.

Całość została zamknięta w funkcji „start()”, która została umieszczona w głównej funkcji „main()”. „start()” zawiera w sobie wywołanie funkcji menu() oraz zbiór instrukcji warunkowych wywołujących dany tryb kalkulatora, w zależności od preferencji użytkownika. Zawartość funkcji „start()” została umieszczona w nieskończonej pętli „while”, powodującej to, że po każdym wykonaniu operacji program wraca do menu i jest gotowy do dalszych aktywności.

6.2 Opis działania program

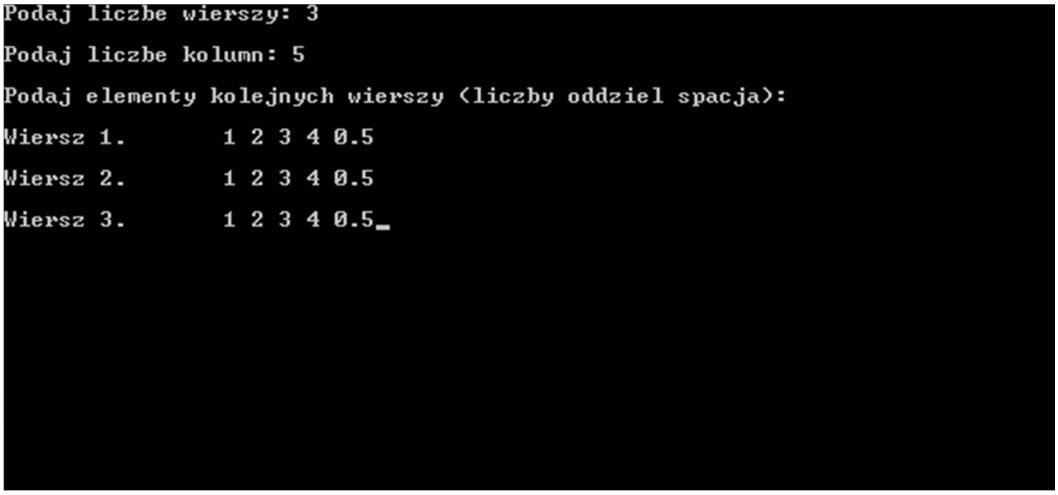
Pogram po uruchomieniu wyświetla menu kalkulatora (Rys. 6.2-1). Przedstawiony interfejs prezentuje użytkownikowi 3 typy możliwych operacji:

1. Operacje na jednej macierzy
2. Operacje na dwóch macierzach
3. Tutoriale (Opisy działań operacji)
4. Zakończenie programu



Rys. 6.2-1 Menu kalkulatora.

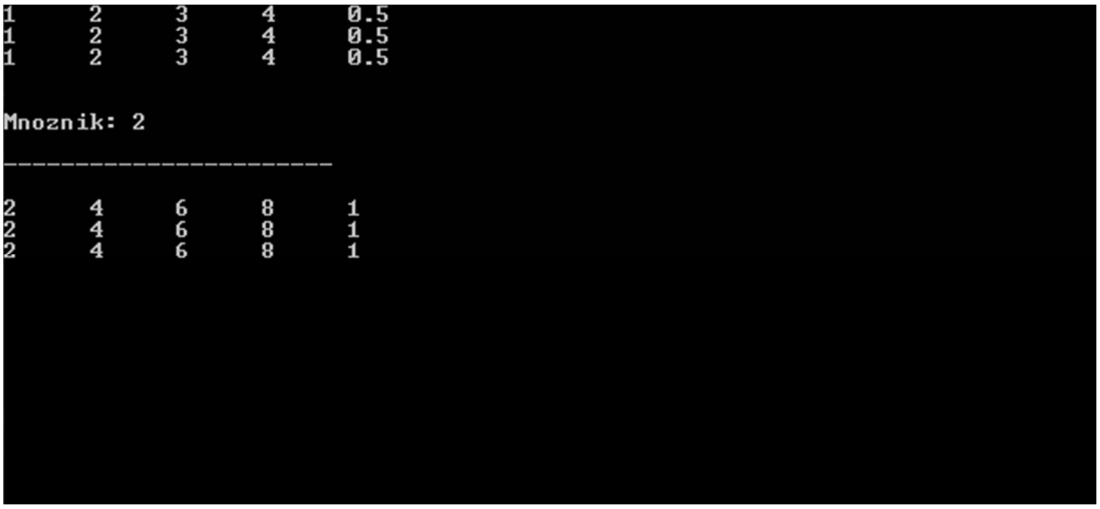
Po wybraniu konkretnej opcji przed użytkownikiem prezentuje się ekran inicjacji macierzy(Rys. 6.2-2). Podawanie macierzy jest zabezpieczone poprzez pętle „do...while” sprawdzające, czy podane macierze spełniają konieczny warunek do wykonania wybranej operacji. W przypadku niepowodzenia, użytkownik zmuszony jest ponownie zainicjować macierz. W przypadku mnożenia macierzy przez liczbę. Zostaje też wywołana prośba o podanie mnożnika (Rys.6.2-3).



Rys. 6.2-2 Okno inicjacji macierzy.

Rys. 6.2-3 Prośba o podanie mnożnika.

Na sam koniec zaprezentowane są rezultaty danej operacji(Rys. 6.2-4). Po wciśnieciu klawisza ENTER, użytkownik zostaje przeniesiony do menu głównego, gdzie ponownie może dokonać wyboru operacji.



Rys. 6.2-4 Prezentacja wyniku.

7. OPIS WYKONANYCH TESTÓW

Kod usterki	Data	Autor	Opis	Stan
Brak kodu	4.01.2018	M. Rafałowski	Podawanie elementów przez użytkownika jest niepoprawne	Naprawiony
Brak kodu	1.02.2018	M. Rafałowski	Złe mnożenie macierzy	Naprawiony

Pierwszym testem było sprawdzenie czy po podaniu odpowiednich rozmiarów macierz zostanie utworzona i wypisana prawidłowo. Macierz wypełniona losowo wypisywała się prawidłowo, kiedy to macierz wypełniona podanymi na wejście elementami powodowała błąd programu. Koniecznym zostało zmodyfikowanie algorytmu wpisywania liczb do macierzy znajdującego się w metodzie set_elem(int w, int k). Bezpośrednie przypisanie każdego elementy zostalo zmienione na wypełnianie pomocniczego vectora, któ®y następnie zostaje wpisany do wektora głównego.

```
double s;
for (int i = 0; i < w; i++)                    //algorytm wpisywania przed poprawką
{
        for (int j = 0; j < k; j++)
        {
                cin>>s;
                matrix[i][j]=s;
        }
}
```

```
double s;
for (int i = 0; i < w; i++)                    //algorytm wpisywania po poprawką
{
        vector<double>temp;

        for (int j = 0; j < k; j++)
        {
                cin>>s;
                temp.push_back(s)
        }

        P3.push_back(temp);
}
```

Podstawowym rodzajem testu wykonanym przeze mnie było porównanie swoich osobistych obliczeń z obliczeniami programu.

7.1 DODAWANIE:

$$A = \begin{matrix} -3 & -0.7 \\ 212 & 8 \\ -1111 & 3{,}9 \end{matrix} \qquad B = \begin{matrix} 11 & 1213 \\ -0.7 & -500 \\ 4 & -5 \end{matrix} \qquad SUMA = \begin{matrix} 8 & 1212{,}3 \\ 211{,}3 & -492 \\ 1107 & -1{,}1 \end{matrix}$$

Test operacji dodawania przebiegł pomyślnie – otrzymany wynik pokrył się z własnoręcznie obliczonym(Rys. 7.1-1).

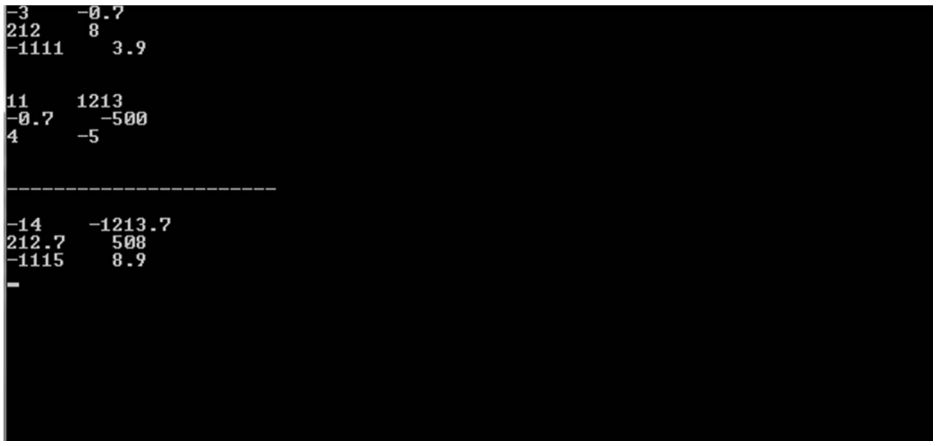
Rys. 7.1-1 Operacja dodawania wykonana w programie.

Drugim testem funkcji dodawania było podanie innych wartości macierzy B sprzecznych z warunkiem dodawania, jakim są identyczne rozmiary obu macierzy. Napisana pętla do while wykonuje się poprawnie, co zmusza użytkownika do ponownego wpisania współczynników zgodnych z założeniami.

7.2 ODEJMOWANIE:

$$A = \begin{matrix} -3 & -0.7 \\ 212 & 8 \\ -1111 & 3{,}9 \end{matrix} \qquad B = \begin{matrix} 11 & 1213 \\ -0.7 & -500 \\ 4 & -5 \end{matrix} \qquad \text{RÓŻNICA} = \begin{matrix} -14 & -1213{,}7 \\ 212{,}7 & 508 \\ -1115 & 8{,}9 \end{matrix}$$

Test operacji dodawania przebiegł pomyślnie – otrzymany wynik pokrył się z własnoręcznie obliczonym (Rys. 7.2-1).



Rys. 7.2-1 Operacja odejmowania wykonana w programie.

Drugim testem funkcji odejmowania było podanie innych wartości macierzy B sprzecznych z warunkiem dodawania, jakim są identyczne rozmiary obu macierzy. Napisana pętla do while wykonuje się poprawnie, co zmusza użytkownika do ponownego wpisania współczynników zgodnych z założeniami.

7.3 MNOŻENIE MACIERZY PRZEZ MACIERZ:

$$A = \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{matrix} \qquad B = \begin{matrix} 1 & 2 \\ 5 & 6 \\ 8 & 9 \end{matrix} \qquad \text{Iloczyn macierzy} = \begin{matrix} 35 & 41 \\ 77 & 92 \end{matrix}$$

Pierwszy test operacji mnożenia zakończył się niepowodzeniem i spowodowaniem błędu „vector subscript out of range". Zdebuggowanie funkcji mnożenia doprowadziło do rozwiązania w postaci zmienienia warunków kończących we wszystkich 3 pętlach.

```
for (int i = 0; i < get_k(); i++)                    //algorytm mnozenia przed poprawką
{
	vector<double>temp;

	for (int j = 0; j < p2.get_w(); j++)
	{
		int sum = 0;

		for (int t = 0; t < get_w(); t++)
		{
			sum += (get_m()[i][t] * p2.get_m()[t][j]);
		}
		p3.set_value(i, j, sum);
	}
}
```

```
for (int i = 0; i < get_w(); i++)                    //algorytm mnozenia po poprawce
{
	vector<double>temp;

	for (int j = 0; j < p2.get_k(); j++)
	{
		int sum = 0;

		for (int t = 0; t < p2.get_w(); t++)
		{
			sum += (get_m()[i][t] * p2.get_m()[t][j]);
		}
		p3.set_value(i, j, sum);
	}
}
```

Po poprawkach w kodzie test operacji operacji mnożenia przebiegł pomyślnie – otrzymany wynik pokrył się z własnoręcznie obliczonym (Rys. 7.3-1).

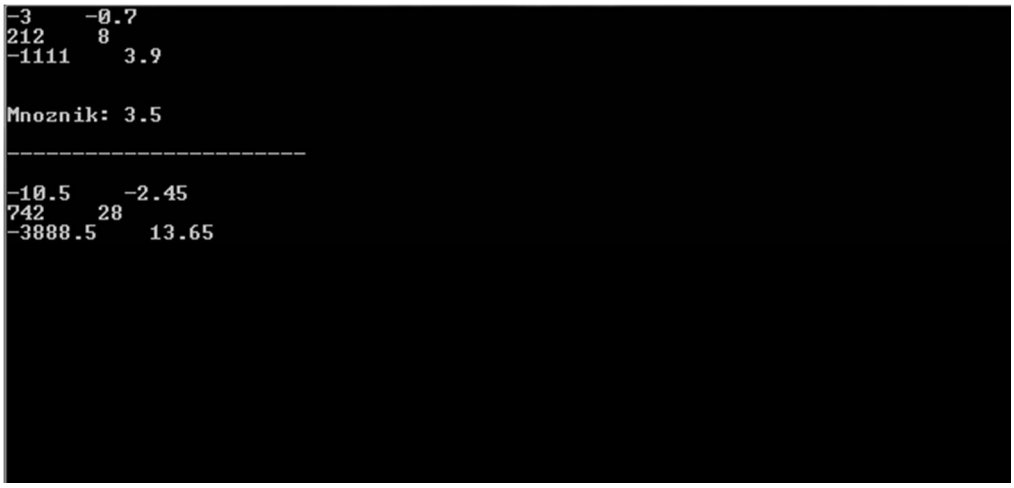
Rys. 7.3-1 Operacja mnożenia macierzy przez macierz wykonana w programie.

Test kontroli odpowiednich współczynników obu macierzy rónież przebiegł pomyślnie.

7.4 MNOŻENIE MACIERZY PRZEZ LICZBĘ:

$$A = \begin{matrix} -3 & -0.7 \\ 212 & 8 \\ -1111 & 3{,}9 \end{matrix} \qquad X = 3{,}5 \qquad \text{ILOCZYN} = \begin{matrix} -10.5 & -2.45 \\ 742 & 28 \\ -3888.5 & 13.65 \end{matrix}$$

Test przebiegł pomyślnie (Rys. 7.4-1).



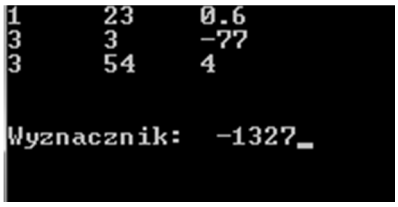
Rys. 7.4-1 Operacja mnożenia przez liczbę wykonana w programie.

7.5 WYZNACZNIK MACIERZY:

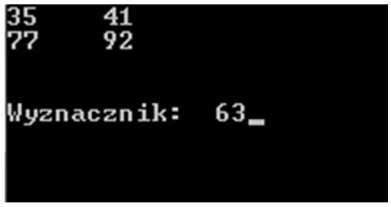
$$A = \begin{matrix} 1 & 23 & 0.6 \\ 3 & 3 & -77 \\ 3 & 54 & 4 \end{matrix} \qquad \det A = -1327$$

$$B = \begin{matrix} 35 & 41 \\ 77 & 92 \end{matrix} \qquad \det B = 63$$

Testy dla obu macierzy przebiegły pomyślnie(Rys. 7.5-1, Rys. 7.5-2).



Rys. 7.5-1 Wyznacznik macierzy A.



Rys. 7.5-2 Wyznacznik macierzy B.

7.6 TRANSPONOWANIE:

Transponowanie macierzy przebiegło pomyślnie (Rys.7.6-1).

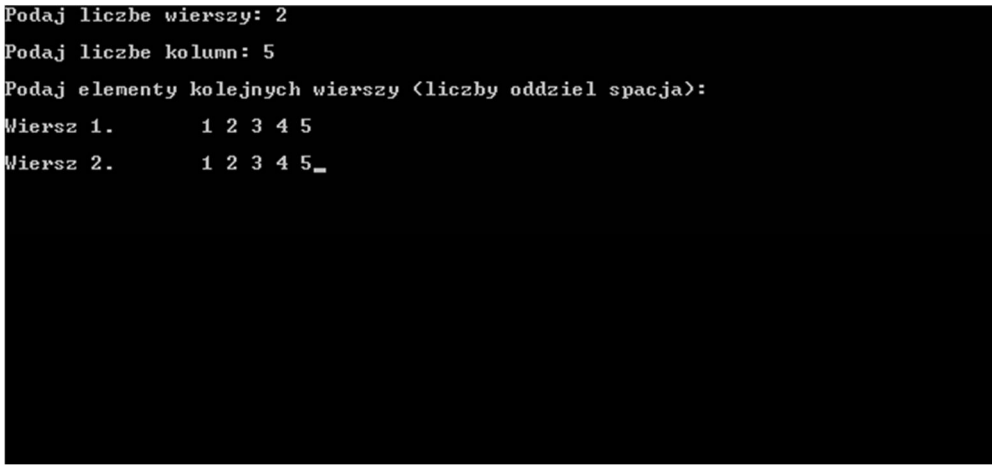
Rys. 7.6-1 Transponowanie podanej macierzy.

8. Podręcznik użytkownika

Użytkownik za pomocą klawiatury numerycznej podaje liczbę odpowiadającą danej operacji kalkulatora, po czym należy wcisnąć ENTER (Rys. 8-1).

Rys. 8-1 Menu kalkulatora.

Następnie zostaje wywołane okno inicjacji macierzy (Rys. 8-2).. Należy wtedy podać wymiary macierzy i elementy kolejnych wierszy oddzielone spacją.



Rys. 8-2 Ekran kreacji macierzy.

Na koniec prezententowany jest wynik danej operacji (Rys. 8-3).

Rys. 8-3 Prezentacja wyniku.

9. Metodologia rozwoju

Program jak najbardziej nadaje się do dalszego wspierania i ulepszania, czy to pod względem nowych funkcji, ulepszonych algorytmów do wcześniej już napisanych działań, czy też poprawek w interfejsie.

W przypadku dodania nowego typu działania należy pamiętać o przyporządkowaniu funkcji własnego unikatowego numeru oraz na wstawieniu nowej instrukcji warunkowej do metody „start()".

10. Bibliografia

[1] *Mnożenie macierzy*, https://pl.wikipedia.org/wiki/Mno%C5%BCenie_macierzy, [Dostęp: 1.02.2018]

[2] *Wyznacznik*, https://pl.wikipedia.org/wiki/Wyznacznik, [Dostęp: 1.02.2018]

[3] *Powtórzenie z Algebry*, http://snauka.pl/powtrzenie-z-algebry.html, [Dostęp: 1.02.2018]

[4] *Metoda Sarrusa dla wyznacznika macierzy wymiaru 3x3*, http://www.dailymotion.com/video/xpgbht, [Dostęp: 1.02.2018]