Graficki prikaz operacija s matricama u programskom jeziku C#

Karajko, Mario

Undergraduate thesis / Završni rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:452929

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-11**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

GRAFIČKI PRIKAZ OPERACIJA S MATRICAMA U PROGRAMSKOM JEZIKU C#

Završni rad

Mario Karajko

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 28.08.2023.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Mario Karajko
Studij, smjer:	Sveučilišni prijediplomski studij Elektrotehnika i informacijska tehnologija
Mat. br. Pristupnika, godina upisa:	4922, 25.09.2020.
OIB Pristupnika:	66052476184
Mentor:	doc. dr. sc. Tomislav Rudec
Sumentor:	izv. prof. dr. sc. Alfonzo Baumgartner
Sumentor iz tvrtke:	
Naslov završnog rada:	Grafički prikaz operacija s matricama u programskom jeziku C#
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada:	Student će napraviti sučelje za upis matrica i grafičko rješenje prikaza operacija s matricama. Tema je zauzeta za studenta: Mario Karajko Sumentor s FERIT-a: Alfonzo Baumgartner
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	28.08.2023.
Datum potvrde ocjene od strane Odbora:	08.09.2023.

Potvrda mentora o predaji konačne verzije rada:	Mentor elektronički potpisao predaju konačne verzije.
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 08.09.2023.

Ime i prezime studenta:	Mario Karajko
Studij:	Sveučilišni prijediplomski studij Elektrotehnika i informacijska tehnologija
Mat. br. studenta, godina upisa:	4922, 25.09.2020.
Turnitin podudaranje [%]:	15

Ovom izjavom izjavljujem da je rad pod nazivom**: Grafički prikaz operacija s matricama u programskom jeziku C#**

izrađen pod vodstvom mentora doc. dr. sc. Tomislav Rudec

i sumentora izv. prof. dr. sc. Alfonzo Baumgartner

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD 1

1.1. Zadatak završnog rada 1

2. PREGLED PODRUČJA TEME 2

3. TEORIJSKA PODLOGA 3

3.1. Definicija matrice 3

3.2. Operacije s matricama 5

3.2.1. Zbrajanje matrica 5

3.2.2. Oduzimanje matrica 5

3.2.3. Množenje matrica 6

3.2.4. Množenje matrice sa skalarom 7

3.3. Microsoft Visual Studio 8

3.4. Programski jezik C# 9

3.5. Windows forme 10

3.5.1. Ukratko o Windows formama 10

3.5.2. Kreiranje Windows forme 10

3.5.3. Važni dijelovi unutar Visual Studio aplikacije 13

4. IZRADA APLIKACIJE 15

4.1. Početna forma 15

4.2. Forma za izbor dimenzija matrica A i B za operacije zbrajanja, oduzimanja i množenja 19

4.3. Forma za izračun rezultantne matrice za operacije zbrajanja, oduzimanja i množenja 22

4.4. Forma za izbor dimenzija matrice i skalara za operaciju množenja sa skalarom 30

4.5. Forma za izračun rezultantne matrice za operaciju množenja sa skalarom 33

5. IZGLED APLIKACIJE 39

6. ZAKLJUČAK 45

LITERATURA 46

SAŽETAK 47

SUMMARY 48

1. UVOD

Matrice predstavljaju poprilično napredan te relativno moderan dio matematike. Osim u matematici, svoju primjenu uz neke ostale grane pronalaze i u računalnoj grafici gdje se koriste za projiciranje trodimenzionalnih slika na dvodimenzionalni ekran.

Tema ovog završnog rada jest izrada Windows desktop aplikacije koja će predstavljati grafički prikaz operacija s matricama u programskom jeziku C# koristeći Windows forme.

Aplikacija je namijenjena svim korisnicima kojima je potreban rad s operacijama zbrajanja, oduzimanja, množenja matrica te množenja matrice sa skalarom, a za izradu same aplikacije korišten je programski jezik C# unutar *Microsoft Visual Studio* razvojnog okruženja te Windows forme kao dio Microsoft .NET platforme.

Korisniku je nakon odabira željene operacije omogućen upis dimenzija matrica ili upis dimenzija matrice i skalara u slučaju odabira operacije množenja matrice sa skalarom. Maksimalna dozvoljena dimenzija matrice ograničena je na dimenziju 8 x 8. Nakon toga, od korisnika se očekuje popunjavanje matrica odnosno matrice cijelim ili decimalnim brojevima te konačno izvršavanje početno odabrane operacije. Također, postoji mogućnost vraćanja matrica i ostalih elemenata na početno stanje te mogućnost povratka na prethodni ili početni prozor.

Strukturu završnog rada čine uvod, teorijska podloga o matricama nakon čega slijedi opis integriranog razvojnog okruženja unutar kojeg su svrstana potpoglavlja o *Microsoft Visual Studio* aplikaciji, programskom jeziku C# te Windows formama. Zatim slijedi pregled područja teme te detaljni opis izrade aplikacije. Prije samog zaključka prikazan je izgled konačne aplikacije, odnosno primjer korištenja aplikacije.

Pri izradi ove aplikacije bitno su pomogla znanja stečena na kolegijima Objektno orijentirano programiranje, Programiranje 1, Programiranje 2 te Linearna algebra.

1.1. Zadatak završnog rada

Zadatak završnog rada jest napraviti Windows desktop aplikaciju koja korisniku omogućuje rad s operacijama s matricama, a to su: zbrajanje, oduzimanje, množenje i množenje sa skalarom.

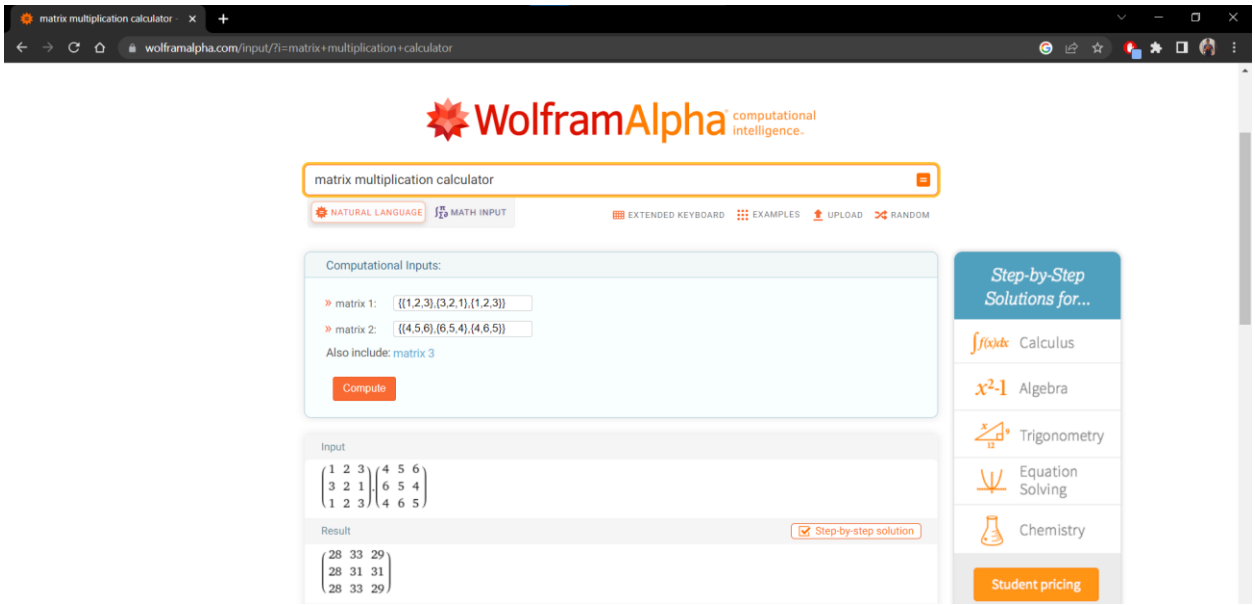
2. PREGLED PODRUČJA TEME

Postoje mnogi dostupni „*online*“ kalkulatori te mobilne aplikacije koje omogućuju izvođenje operacija s matricama.

Primjer „*online*“ kalkulatora može se pronaći na poveznici „*https://www.wolframalpha.com/input/?i=matrix+multiplication+calculator*“ koja nudi mogućnost izvođenja operacije množenja matrica, ali za detaljniji prikaz korak-po-korak rješenja potrebno je platiti mjesečnu pretplatu kako bi se otključala ta mogućnost. Drugi primjer „*online*“ kalkulatora može se pronaći na poveznici „*https://matrix.reshish.com/multiplication.php*“, a nudi mogućnost izvođenja operacija zbrajanja, oduzimanja i množenja matrica uz objašnjenje uvjeta za izvođenje operacije i napomene vezane za istu. Nedostatak ovakvih „*online*“ kalkulatora je taj da mora biti ostvaren pristup internetskoj vezi kako bi korisnik uopće mogao pristupiti traženoj internetskoj stranici.

Primjer mobilne aplikacije jest aplikacija „*Symbolab*“ koja nudi mogućnost izvođenja raznih operacija s matricama pružajući pri tome i detaljniji prikaz izvođenja te operacije s time da se korisnik također može i pretplatiti na „*Symbolab Pro*“ verziju koja pruža još više mogućnosti poput punog pristupa koracima rješenja, bez reklama i pristup s bilo kojeg uređaja.

Na slici 2.1. može se vidjeti rezultat izvođenja operacije množenja matrica na internetskoj stranici „*WolframAlpha*“.



Slika 2.1. *WolframAlpha* „*online*“ kalkulator

3. TEORIJSKA PODLOGA

3.1. Definicija matrice

Matrica tipa (*m, n*) s koeficijentima iz polja $\mathbb{F}$, pri čemu su *m* i *n* prirodni brojevi, definira se kao preslikavanje:

$$A:\{1,2,3,\ldots,m\}\times\{1,2,3,\ldots,n\}\to \mathrm{F}. \tag{3-1}$$

Pod poljem $\mathbb{F}$ najčešće smatramo polje realnih brojeva $\mathbb{R}$ ili polje kompleksnih brojeva $\mathbb{C}$. Djelovanje funkcije A često se daje u tabličnom obliku koji se sastoji od *m* redaka i *n* stupaca. Pri tome se kaže da *i*-ti redak i *j*-ti stupac ima funkcijsku vrijednost $A(i, j)$ ili jednostavnije a_{ij}. Sukladno ovim dogovorima, svaka matrica koja se sastoji od *m* redaka i *n* stupaca može se zapisati u tabličnom obliku:

$$A=\left[a_{ij}\right]=\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}. \tag{3-2}$$

Skup svih matrica koje se sastoje od *m* redaka i *n* stupaca s koeficijentima iz polja $\mathbb{F}$ označavamo s $M_{mn}(\mathbb{F})$. Ukoliko je broj redaka jednak broju stupaca, odnosno $m = n$, kraće pišemo $M_n(\mathbb{F})$, a za elemente tog skupa kažemo da su kvadratne matrice reda *n* [1].

Za matrice $A = [a_{ij}] \in M_{mn}$ i $B = [b_{ij}] \in M_{mn}$ s *m* redaka i *n* stupaca kažemo da su jednake ako su istog tipa, odnosno ako imaju jednak broj redaka i stupaca te ako imaju jednake odgovarajuće elemente [2], tj. ako vrijedi:

$$a_{ij}=b_{ij},\quad \forall i=1,2,3,\ldots,m,\quad \forall j=1,2,3,\ldots,n. \tag{3-3}$$

Također, vrijedi spomenuti pojmove dijagonalne, skalarne, jedinične, pravokutne te nul-matrice.

Dijagonalna matrica jest kvadratna matrica kojoj su svi elementi, koji nisu na glavnoj dijagonali, jednaki nuli, tj. vrijedi:

$$a_{ij} = 0, \quad \forall i, j = 1, 2, 3, \ldots, n, \quad i \neq j. \tag{3-4}$$

U nastavku je prikazan primjer dijagonalne matrice:

$$A = \begin{bmatrix} 7 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 3 \end{bmatrix}.$$

Skalarna matrica jest dijagonalna matrica kojoj su svi elementi na glavnoj dijagonali jednaki. Primjer skalarne matrice prikazan je u nastavku:

$$A = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix}.$$

Jedinična matrica jest skalarna matrica kojoj su svi elementi na glavnoj dijagonali jednaki 1, tj. vrijedi:

$$a_{11} = a_{22} = a_{33} = \ldots = a_{nn} = 1. \tag{3-5}$$

Uobičajena oznaka jedinične matrice s *n* redaka i *n* stupaca jest I_n. U nastavku je prikazan primjer jedinične matrice:

$$I_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Pod pojmom „pravokutna“ matrica smatramo matricu kojoj broj redaka nije jednak broju stupaca. Primjer takve matrice prikazan je u nastavku:

$$A = \begin{bmatrix} 4 & 0 & 8 & -3 \\ 1 & -9 & 2 & 5 \end{bmatrix}.$$

Nul-matrica jest matrica kojoj su svi elementi jednaki nuli, tj. vrijedi: $a_{ij} = 0, \ \forall i, j.$ (3-6)

3.2. Operacije s matricama

3.2.1. Zbrajanje matrica

Operacija zbrajanja matrica definirana je ako i samo ako su matrice istog tipa.

Neka su $A = [a_{ij}] \in M_{mn}$ te $B = [b_{ij}] \in M_{mn}$.

Zbroj matrica A i B jest matrica $C := A + B \in M_{mn}$ takva da vrijedi:

$$c_{ij} = a_{ij} + b_{ij}, \quad \forall i = 1, 2, 3, \ldots, m, \quad \forall j = 1, 2, 3, \ldots, n. \tag{3-7}$$

U tabličnom obliku operacija zbrajanja matrica izgleda ovako:

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \ldots & b_{1n} \\ b_{21} & b_{22} & \ldots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \ldots & b_{mn} \end{bmatrix} = \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \ldots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \ldots & a_{2n}+b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \ldots & a_{mn}+b_{mn} \end{bmatrix}. \tag{3-8}$$

U nastavku je prikazan primjer zbrajanja dvije matrice:

$$\begin{bmatrix} -5 & 1 & 4 \\ 4 & 7 & -1 \\ 2 & 0 & 9 \end{bmatrix} + \begin{bmatrix} 8 & 7 & 2 \\ 1 & 3 & 2 \\ 0 & -2 & -3 \end{bmatrix} = \begin{bmatrix} 3 & 8 & 6 \\ 5 & 10 & 1 \\ 2 & -2 & 6 \end{bmatrix}.$$

3.2.2. Oduzimanje matrica

Operacija oduzimanja matrica se analogno definira te matrice također moraju biti istog tipa.

Neka su $A = [a_{ij}] \in M_{mn}$ te $B = [b_{ij}] \in M_{mn}$.

Razlika matrica A i B jest matrica $C := A - B \in M_{mn}$ takva da vrijedi:

$$c_{ij} = a_{ij} - b_{ij}, \quad \forall i = 1, 2, 3, \ldots, m, \quad \forall j = 1, 2, 3, \ldots, n. \tag{3-9}$$

U tabličnom obliku operacija oduzimanja matrica izgleda ovako:

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix} - \begin{bmatrix} b_{11} & b_{12} & \ldots & b_{1n} \\ b_{21} & b_{22} & \ldots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \ldots & b_{mn} \end{bmatrix} = \begin{bmatrix} a_{11}-b_{11} & a_{12}-b_{12} & \ldots & a_{1n}-b_{1n} \\ a_{21}-b_{21} & a_{22}-b_{22} & \ldots & a_{2n}-b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}-b_{m1} & a_{m2}-b_{m2} & \ldots & a_{mn}-b_{mn} \end{bmatrix}. \tag{3-10}$$

U nastavku je prikazan primjer oduzimanja dvije matrice:

$$\begin{bmatrix} 2 & 9 \\ -7 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 2 & -5 \end{bmatrix} = \begin{bmatrix} 1 & 9 \\ -9 & 6 \end{bmatrix}.$$

3.2.3. Množenje matrica

Operacija množenja matrica definirana je ako i samo ako su matrice ulančane, tj. broj stupaca prve matrice mora biti jednak broju redaka druge matrice.

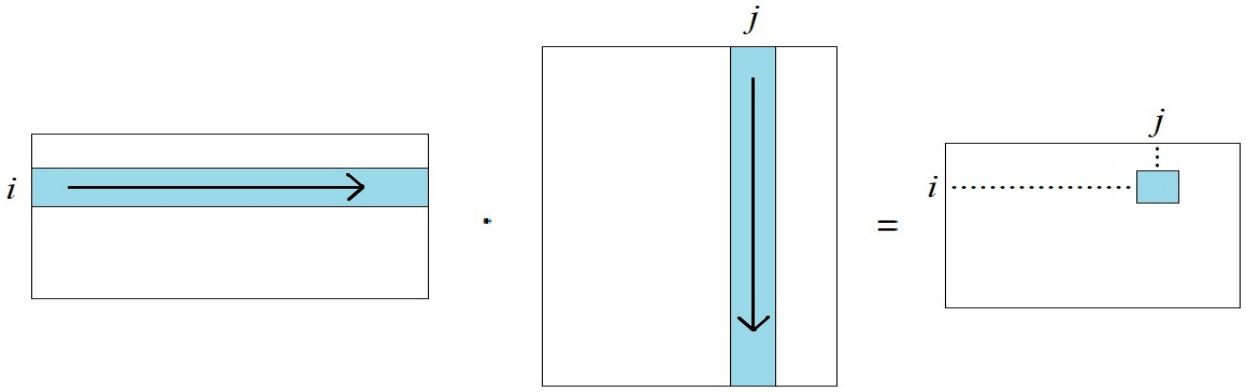
Neka je $A = [a_{ij}] \in M_{mn}$ te $B = [b_{ij}] \in M_{rp}$. Kažemo da su matrice ulančane ako je $n = r$.

Rezultat množenja matrice A i B bit će matrica $C = [c_{ij}] \in M_{mp}$ tipa (m, p) [1].

Opći element umnoška c_{ij} dan je sljedećom formulom:

$$c_{ij} = a_{i1} \cdot b_{1j} + a_{i2} \cdot b_{2j} + \ldots + a_{in} \cdot b_{nj} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}, \quad \forall i = 1, 2, 3, \ldots, m, \quad \forall j = 1, 2, 3, \ldots, p. \quad (3\text{-}11)$$

Grafički se operacija množenja matrica može prikazati na način prikazan na slici 3.1.:



Slika 3.1. Grafički prikaz operacije množenja matrica

Vrijedi još napomenuti da množenje matrica općenito nije komutativno, tj. vrijedi: $A \cdot B \neq B \cdot A$.

U nastavku je prikazan primjer množenja dviju matrice:

$$\begin{bmatrix} 6 & 1 & -3 \\ 2 & 7 & -9 \end{bmatrix} \cdot \begin{bmatrix} 9 & 3 & 4 & 0 \\ 2 & -4 & 1 & 8 \\ 1 & 7 & -5 & -3 \end{bmatrix} = \begin{bmatrix} 53 & -7 & 40 & 17 \\ 23 & -85 & 60 & 83 \end{bmatrix}.$$

3.2.4. Množenje matrice sa skalarom

Neka je $\lambda \in \mathbb{R}$ bilo koji skalar, te $A = [a_{ij}] \in M_{mn}$. Umnožak matrice A sa skalarom λ jest matrica λA definirana na sljedeći način:

$$\lambda \left[a_{ij} \right] = \left[\lambda a_{ij} \right]. \tag{3-12}$$

Može se zaključiti da se matrica množi sa skalarom na način da se svaki njezin element pomnoži s tim skalarom.

U tabličnom obliku operacija množenja matrice sa skalarom izgleda ovako:

$$\lambda \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} = \begin{bmatrix} \lambda a_{11} & \lambda a_{12} & \dots & \lambda a_{1n} \\ \lambda a_{21} & \lambda a_{22} & \dots & \lambda a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda a_{m1} & \lambda a_{m2} & \dots & \lambda a_{mn} \end{bmatrix}. \tag{3-13}$$

U nastavku je prikazan primjer množenja matrice sa skalarom:

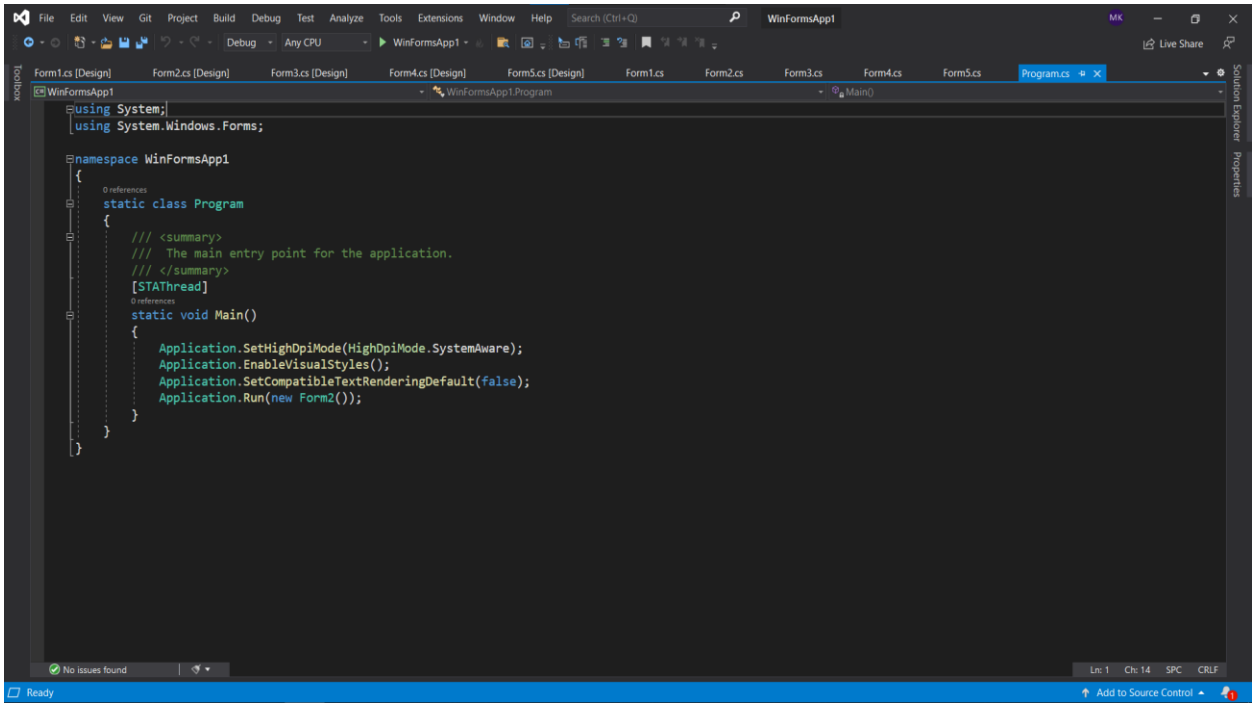
$$-3 \begin{bmatrix} 2 & 7 & -5 & 1 \\ 9 & -2 & 4 & 10 \\ 0 & 1 & -4 & 3 \\ 8 & 5 & 0 & 6 \end{bmatrix} = \begin{bmatrix} -6 & -21 & 15 & -3 \\ -27 & 6 & -12 & -30 \\ 0 & -3 & 12 & -9 \\ -24 & -15 & 0 & -18 \end{bmatrix}.$$

3.3. Microsoft Visual Studio

Microsoft Visual Studio integrirano je razvojno okruženje (engl. *integrated development environment* ili skraćeno *IDE*) američke tvrtke Microsoft koje predstavlja moćan i sveobuhvatan razvojni alat. Može se koristiti za pisanje, uređivanje, debugiranje i izradu koda, a naposljetku i za implementaciju same aplikacije. Osim uređivanja koda i debugiranja, Visual Studio uključuje kompajlere, alate za dovršavanje koda, kontrolu izvora, razna proširenja i mnoge druge značajke za poboljšanje svake faze procesa razvoja samog softvera.

Zbog mnoštva značajki i podrške za programske jezike koje Visual Studio pruža, napredak od pisanja „*Hello World*“ programa do razvoja i implementacije zahtjevnijih aplikacija itekako može biti zanimljiv. Na primjer, moguće je izrađivati, debugirati i testirati .NET i C++ aplikacije, uređivati ASP.NET stranice, razvijati više-platformske mobilne i desktop aplikacije pomoću .NET-a ili izraditi responzivno Web korisničko sučelje u C#-u [3].

Na slici 3.2. može se vidjeti radno okruženje unutar same aplikacije Microsoft Visual Studio.



Slika 3.2. Radno okruženje unutar Microsoft Visual Studio aplikacije

3.4. Programski jezik C#

Programski jezik C# jest moderan jezik koji je u potpunosti objektno orijentiran i neovisan o platformi. C# omogućuje programerima izradu mnogih tipova sigurnih i robusnih aplikacija pokretanih unutar .NET platforme [4].

Sama .NET platforma jest besplatna razvojna platforma otvorenog koda razvijena od strane Microsoft-a koja služi za izradu različitih vrsta aplikacija poput: web i mobilnih aplikacija, desktop aplikacija, igrica, Internet of Things (skraćeno *IoT*) aplikacija i drugih. Same .NET aplikacije mogu se pisati u nekom od sljedećih programskih jezika: #C, F# ili Visual Basic [5].

C# svoje korijene ima unutar C obitelji programskih jezika i odmah će biti poznat C, C++, Java i JavaScript programerima [4]. Budući da C# pripada pod objektno orijentirane jezike, potrebno je reći da sama objektno orijentirana paradigma počiva na tri osnovna koncepta (*tzv*. stupovi objektno orijentiranog programiranja):

1. Enkapsulaciji (*tzv*. učahurivanje) – označava ideju da klasa unutar sebe ugrađuje stanje i ponašanje, a prema vanjskom svijetu otvara samo ono što je apsolutno nužno za njene korisnike.
2. Hijerarhijskom odnosu (specijalizacija i generalizacija) među klasama postignutim nasljeđivanjem – prilikom nasljeđivanja jedna se klasa izvodi iz druge pri čemu je ta izvedena klasa zapravo podvrsta osnovne klase i posjeduje sva obilježja (stanje i ponašanje) osnovne klase. Također joj je omogućena njihova eventualna prilagodba vlastitim specifičnim potrebama, ali i dodavanje novog stanja i ponašanja.
3. Polimorfizmu - leži u ideji da je moguće isto sučelje iskoristiti za više različitih klasa, pristupati objektima tih klasa preko zajedničkog sučelja, a u isto vrijeme očekivati njihovo specijalizirano ponašanje.

Programski jezik C# smatra se odličnim izborom za kreiranje Windows desktop aplikacija. Bazira se na objektno orijentiranim principima na način da prikuplja podatke u objekte što olakšava razbijanje aplikacije na manje dijelove koji su brži i lakši za izradu, upravljanje i kombiniranje. Smatra se jezikom visoke razine jer ima jednostavnu sintaksu za razumijevanje i upravljanje što nije slučaj s jezicima niske razine kao što je programski jezik C [6].

3.5. Windows forme

3.5.1. Ukratko o Windows formama

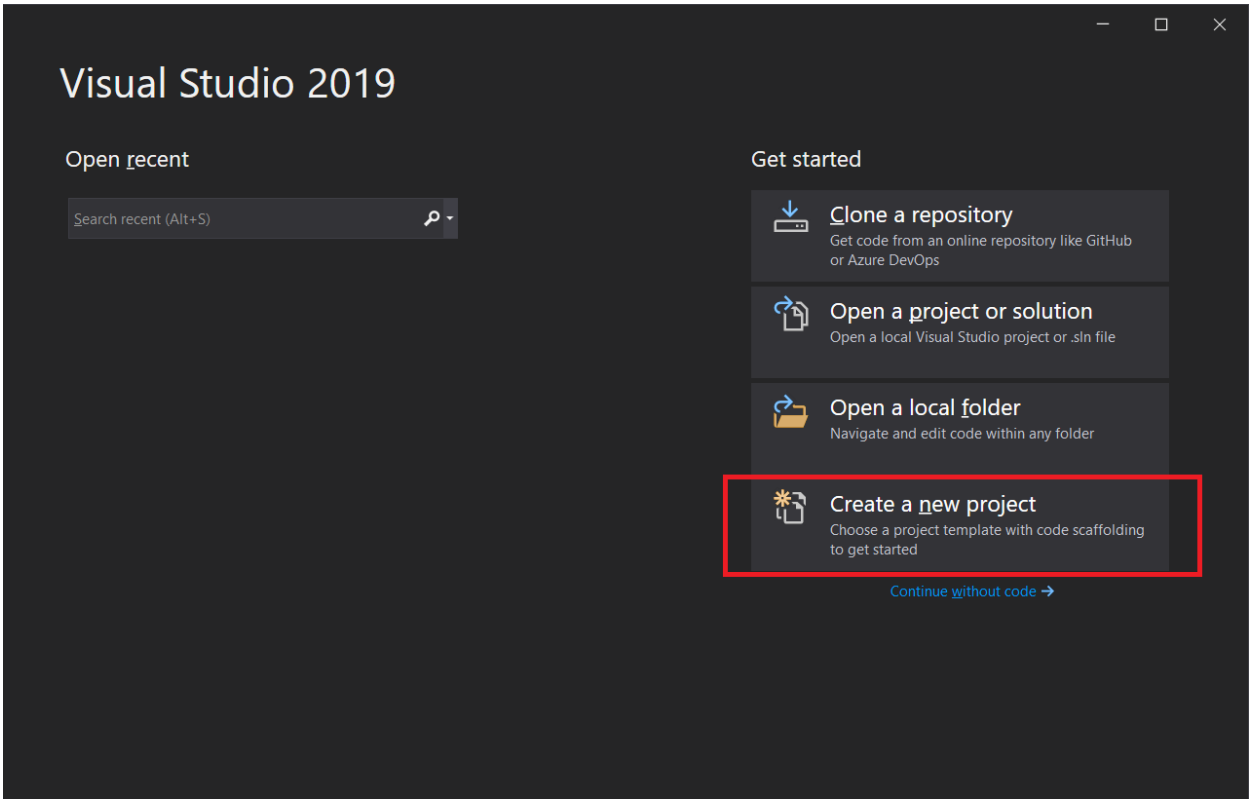
Windows forme (engl. *Windows Forms*) predstavljaju okvir, odnosno „*Framework*“ korisničkog sučelja za izradu Windows desktop aplikacija. Pruža jedan od najproduktivnijih načina za kreiranje desktop aplikacija temeljenih na vizualnom dizajneru kojeg Visual Studio nudi. Funkcionalnost poput „*drag-and-drop*“ postavljanja vizualnih kontrola omogućuje vrlo jednostavnu izradu samih desktop aplikacija. Pomoću Windows formi moguće je razvijati grafički bogate aplikacije koje je lako implementirati, ažurirati te raditi na njima bilo „*offline*“ bilo „*online*“. Windows Forms aplikacije mogu pristupiti lokalnom hardveru i datotečnom sustavu računala na kojemu je aplikacija pokrenuta [7].

3.5.2. Kreiranje Windows forme

Prvi je korak prilikom kreiranja nove aplikacije otvaranje Visual Studio programa te generiranje aplikacije iz predloška. U nastavku su detaljno objašnjeni koraci za kreiranje aplikacije [8]:

1. Otvoriti Visual Studio.
2. Odabrati „*Create a new project*“.

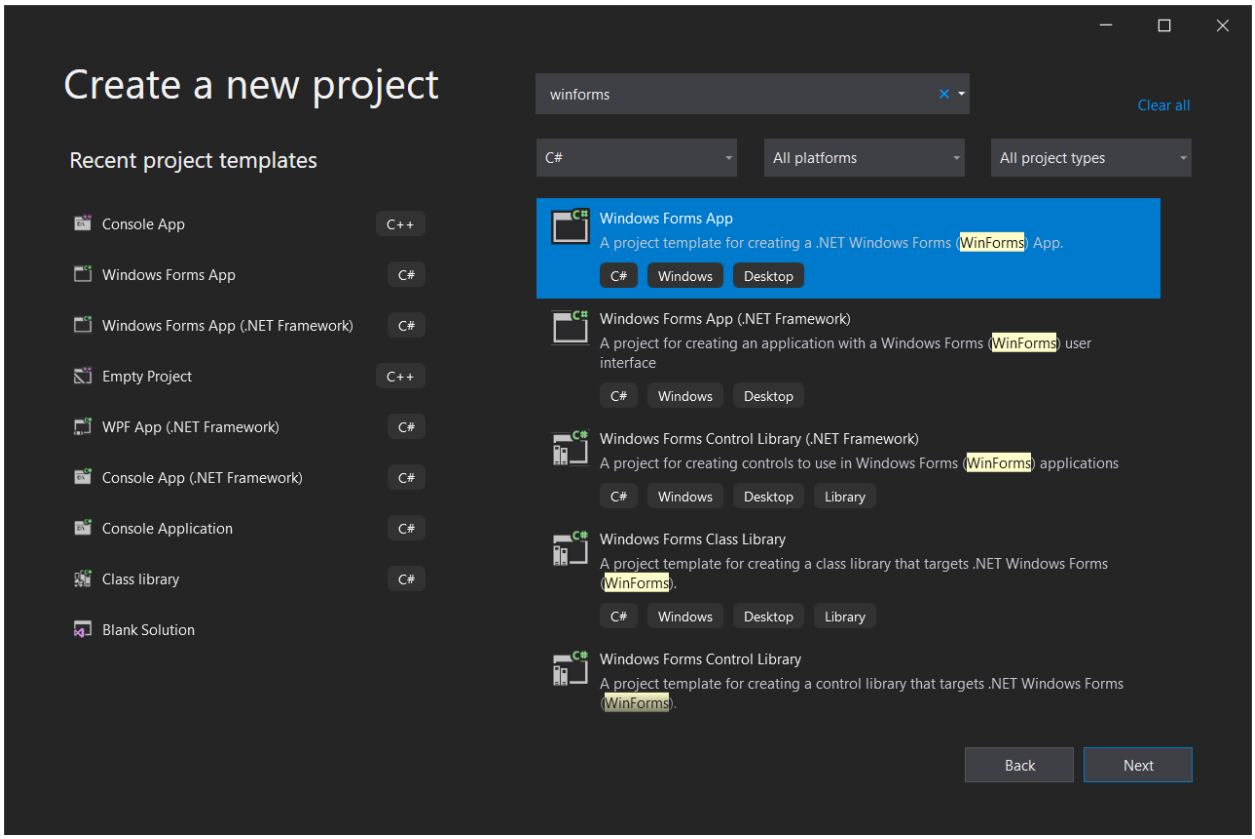
Na slici 3.3. može se vidjeti početni prozor Visual Studio aplikacije.



Slika 3.3. Početni prozor Visual Studio aplikacije

3. Unutar „*Search for templates*“ okvira upisati „*winforms*“ i pričekati da se pojave rezultati pretraživanja.
4. Unutar padajućeg izbornika s jezicima odabrati C#.
5. Unutar popisa predložaka odabrati „*Windows Forms App*“ i kliknuti „*Next*“.

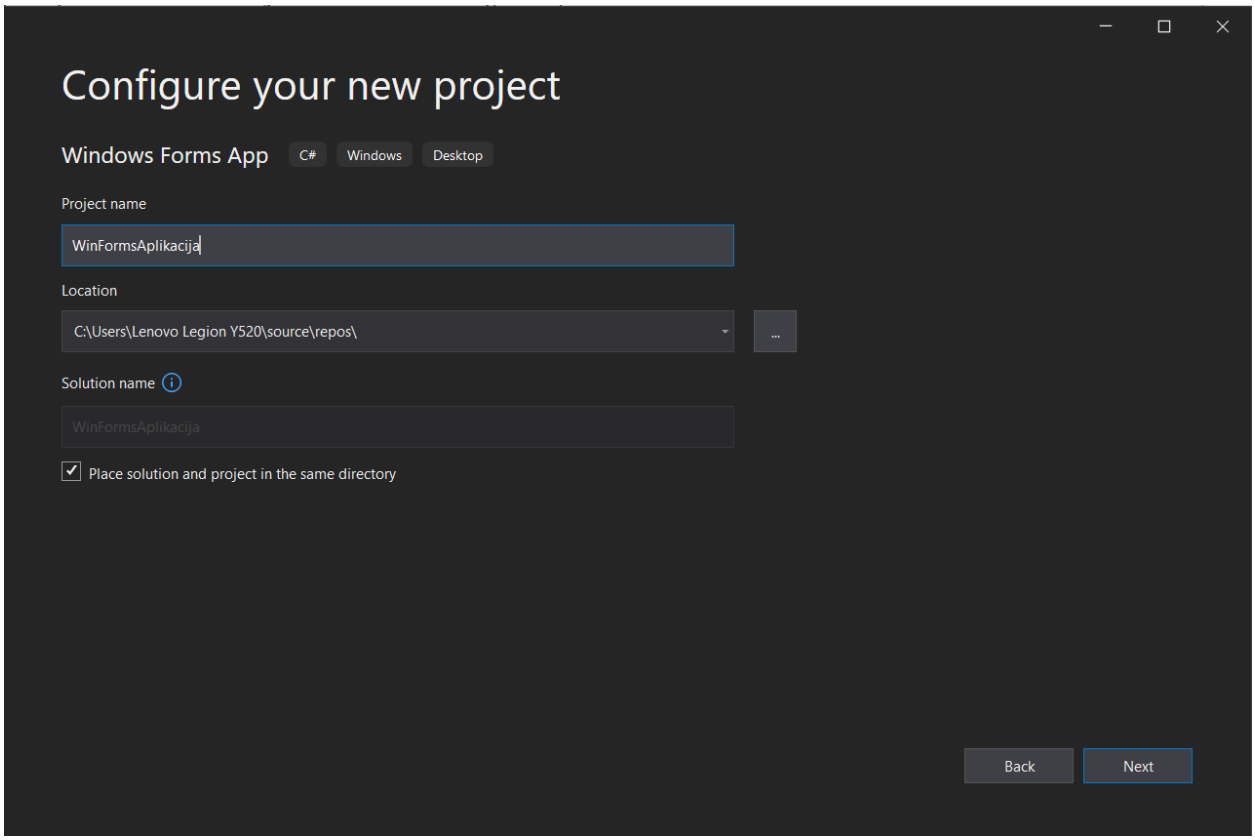
Na slici 3.4. može se vidjeti kako bi trebao izgledati trenutni prozor Visual Studio aplikacije.



Slika 3.4. Prozor Visual Studio aplikacije

6. Unutar „*Configure your new project*“ prozora upisati proizvoljno ime projekta te po potrebi promijeniti mapu gdje će projekt biti spremljen pomoću podešavanja „*Location*“ putanje i kliknuti „*Next*“.

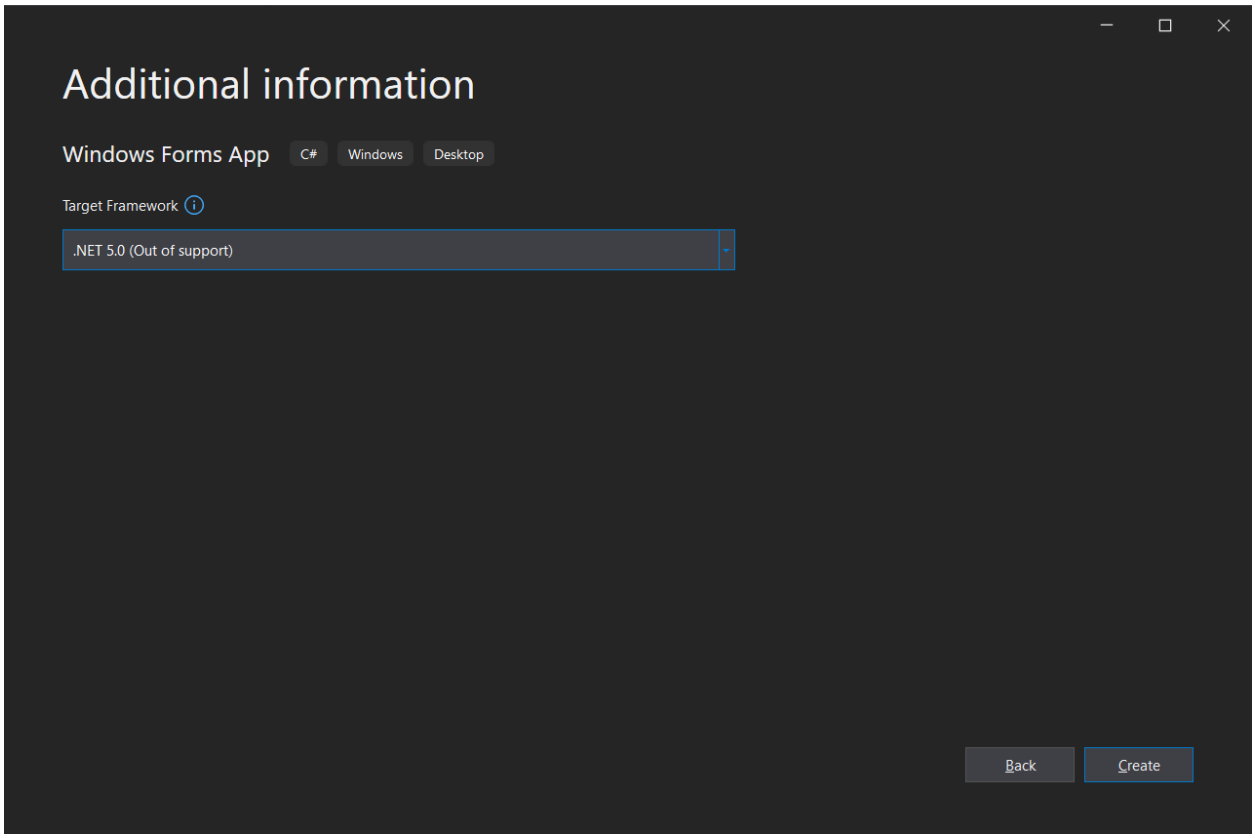
Na slici 3.5. može se vidjeti „*Configure your new project*“ prozor.



Slika 3.5. „*Configure your new project*“ prozor

7. Konačno, unutar „*Additional information*“ prozora odabrati ciljani .NET „*Framework*“ i kliknuti „*Create*“.

Na slici 3.6. može se vidjeti završni „*Additional information*“ prozor.



Slika 3.6. Završni „*Additional infromation*“ prozor

Jednom kada je aplikacija generirana, Visual Studio bi trebao otvoriti dizajnersko okno za zadanu formu – „*Form1“*. Ukoliko dizajn forma nije vidljiva, potrebno je napraviti dvostruki klik na „*Form1.cs*“ unutar Solution Explorer okna kako bi se otvorio dizajn prozor.

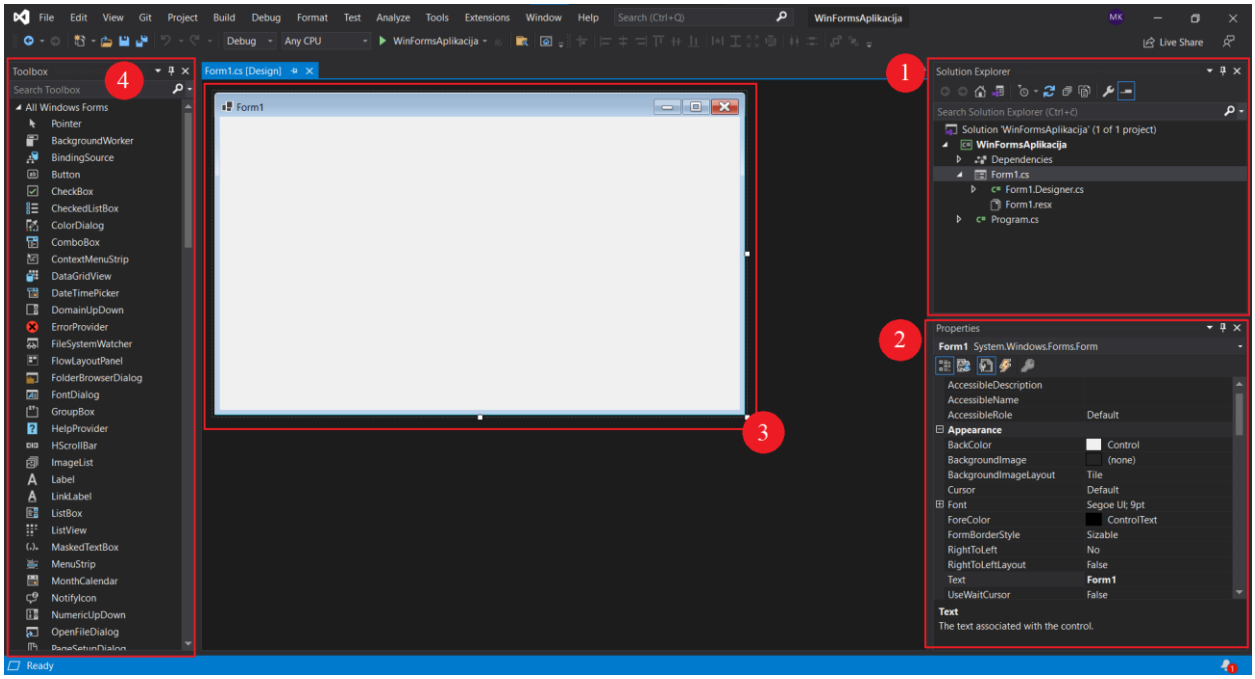
3.5.3. Važni dijelovi unutar Visual Studio aplikacije

Podrška za Windows forme unutar Visual Studio aplikacije sastoji se od četiri važne komponente pomoću kojih se vrši interakcija prilikom stvaranja aplikacije [8]:

1. Solution Explorer – unutar ovoga okna nalaze se sve projektne datoteke, kod, forme i resursi.
2. Properties – ovo okno prikazuje postavke svojstava koje je moguće konfigurirati na temelju odabrane stavke. Na primjer, ukoliko se odabere stavka iz Solution Explorer-a, bit će prikazane postavke svojstava koje se odnose na tu datoteku. Ukoliko se odabere objekt unutar „*Designer-a*“, bit će prikazane postavke vezane za tu kontrolu ili formu.
3. Form Designer – predstavlja alat za dizajniranje, odnosno dizajner za formu. Interaktivan je i moguće je napraviti „*drag-and-drop*“ objekata iz Toolbox-a. Odabirom i premještanjem stavki u dizajneru moguće je vizualno, odnosno grafički sastaviti korisničko sučelje (engl. *UI*) za samu aplikaciju.

4. Toolbox – sadrži sve kontrole koje je moguće dodati u formu. Da bi se kontrola dodala u trenutnu formu potrebno je napraviti dvostruki klik na željenu kontrolu ili napraviti „*drag-and-drop*“ same kontrole u formu.

Na slici 3.7. mogu se vidjeti ove četiri komponente unutar Visual Studio aplikacije.



Slika 3.7. Važne komponente za rad s Windows formama

4. IZRADA APLIKACIJE

Sama aplikacija sastoji se od pet Windows formi te od „*Program.cs*“ datoteke. Svaka će forma biti objašnjena posebno i to redoslijedom kako se sama aplikacija i koristi.

Datoteka „*Program.cs*“ predstavlja ulaznu točku aplikacije, a koja se automatski generira prilikom kreiranja projekta. Sadrži *Main* metodu koja je polazišna točka aplikacije i definira odakle će se program početi izvršavati. U ovome slučaju, aplikaciji je postavljena forma *Form1* kao zadana, odnosno početna forma koja će biti prikazana prilikom pokretanja aplikacije. Programski kod 4.1. ove datoteke dan je u nastavku:

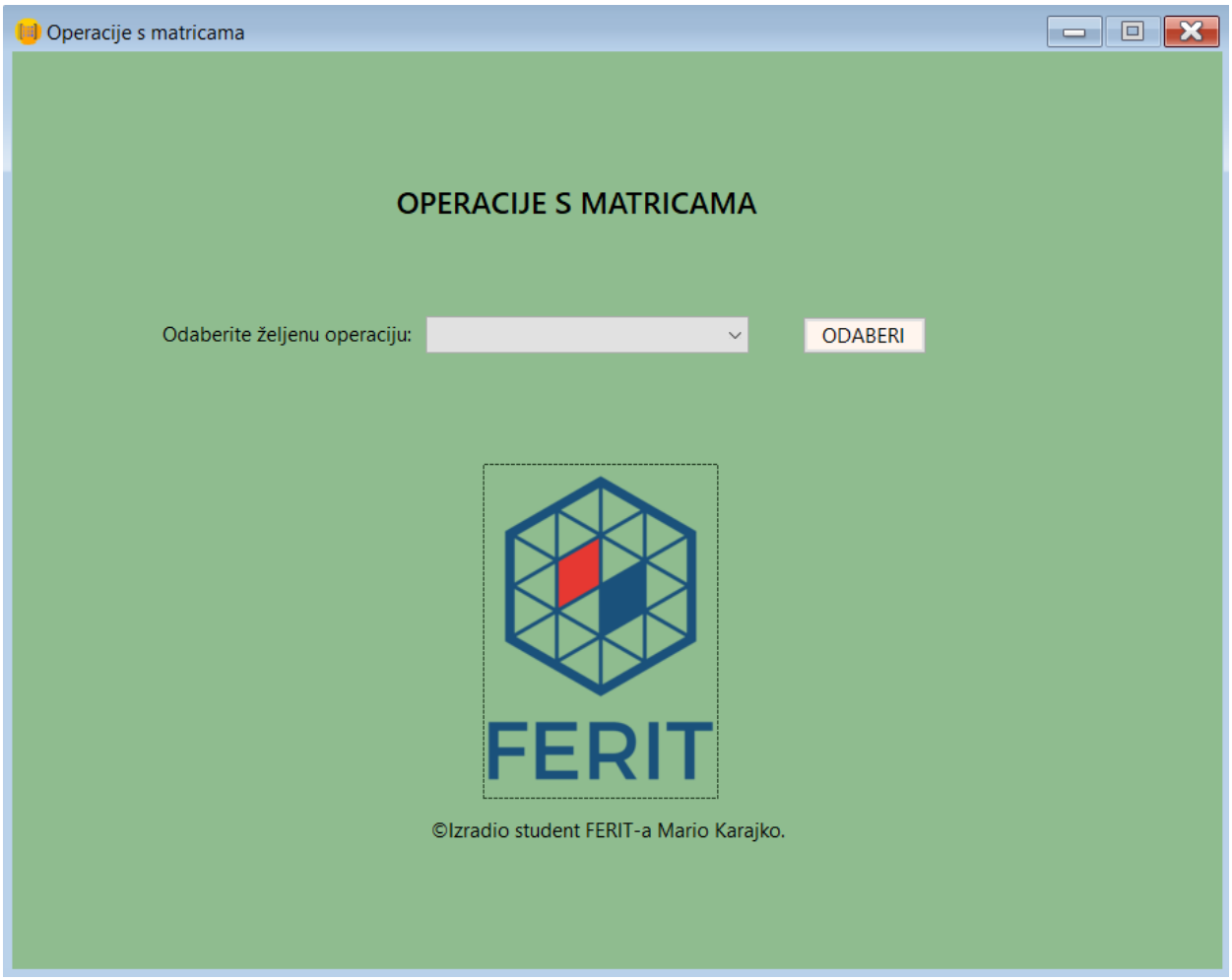
```
static class Program
{
    /// <summary>
    ///  The main entry point for the application.
    /// </summary>
    [STAThread]
    static void Main()
    {
        Application.SetHighDpiMode(HighDpiMode.SystemAware);
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new Form1());
    }
}
```

Programski kod 4.1. Datoteka „*Program.cs*“

4.1. Početna forma

Prva forma imena „*Form1.cs*“ predstavlja početni prozor koji se pojavi kada korisnik pokrene aplikaciju.

Na slici 4.1. može se vidjeti izgled prve, odnosno početne forme.



Slika 4.1. Izgled početne forme *Form1*

Početna se forma sastoji od sljedećih kontrola, odnosno elemenata:

1. Label (hrv. *oznaka*) koji prikazuje tekst „OPERACIJE S MATRICAMA“.
2. Label koji prikazuje tekst „Odaberite željenu operaciju:“.
3. ComboBox imena „izbornikOperacija“ koji predstavlja padajući izbornik koji sadrži kolekciju sljedećih *string*-ova:
    - „Zbrajanje“,
    - „Oduzimanje“,
    - „Množenje“,
    - „Množenje sa skalarom“.
4. Button imena „odaberiGumb“ koji predstavlja gumb za odabir željene operacije.
5. PictureBox koji predstavlja FERIT logo.
6. Label koji prikazuje tekst „©Izradio student FERIT-a Mario Karajko.“.

Dodatno, svaka forma ima zajedničku ikonicu u gornjem lijevom kutu koja je pohranjena lokalno unutar samog projekta. Također, svaka forma sadrži sliku FERIT logotipa i label koji prikazuje tekst „©Izradio student FERIT-a Mario Karajko.“ pa se stoga u daljnjim formama više neće spominjati.

Nakon dijela koji se tiče grafičkog izgleda početne forme, potrebno je objasniti i programski kod koji se krije iza forme.

Programski kod 4.2. prikazuje javnu statičku *string* varijablu „operacija“ koja će biti potrebna u drugim formama kako bi se ispravno prikazao naziv operacije koja će se izvoditi. Nadalje, vidljiv je i konstruktor klase *Form1* koji služi za inicijalizaciju komponenti same forme.

```
public static string operacija = "";
public Form1()
{
        InitializeComponent();
}
```

Programski kod 4.2. Konstruktor i statička *string* varijabla

Sljedeća funkcija, odnosno metoda prikazana u programskom kodu 4.3. predstavlja događaj (engl. *event*) koji se izvršava kada korisnik klikne na gumb imena „odaberiGumb“.

```
private void odaberiGumb_Click(object sender, EventArgs e)
{
    if (izbornikOperacija.Text == "Množenje")
    {
        operacija = "Množenje";
        Hide();
        new Form2().ShowDialog();
        Close();
    }

    else if (izbornikOperacija.Text == "Zbrajanje")
    {
        operacija = "Zbrajanje";
        Hide();
        new Form2().ShowDialog();
        Close();
    }

    else if (izbornikOperacija.Text == "Oduzimanje")
    {
        operacija = "Oduzimanje";
        Hide();
        new Form2().ShowDialog();
        Close();
    }

    else if (izbornikOperacija.Text == "Množenje sa skalarom")
    {
        operacija = "Množenje sa skalarom";
        Hide();
        new Form4().ShowDialog();
        Close();
    }

    else
    {
        MessageBox.Show("Morate odabrati neku operaciju!", "Pogreška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

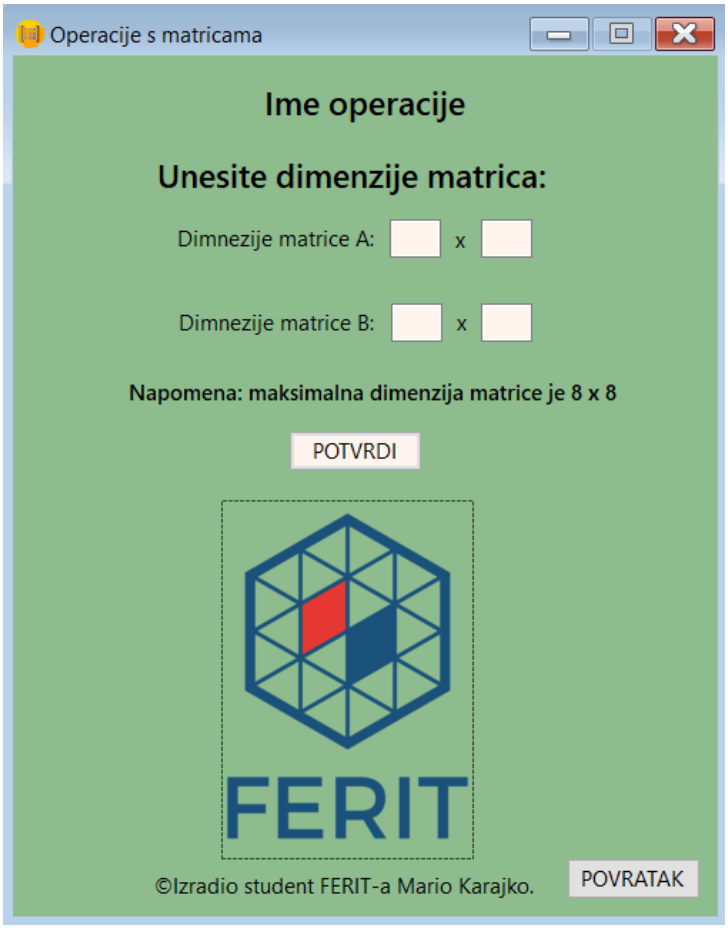
Programski kod 4.3. Događaj na klik gumba „odaberiGumb“

Sam događaj klika na gumb obrađen je na način da ukoliko je korisnik izabrao željenu operaciju unutar padajućeg izbornika, statička *string* varijabla „operacija“ poprimit će naziv odabrane operacije, a trenutna, odnosno početna forma bit će prvo sakrivena, a zatim i zatvorena te će se otvoriti sljedeća forma *Form2* ili *Form4* ovisno već koju je operaciju korisnik odabrao. Ukoliko korisnik nije odabrao niti jednu operaciju od ponuđenih, a kliknuo je na „odaberiGumb“, pojavit će se mali prozorčić s porukom o pogrešci kako mora odabrati neku operaciju.

4.2. Forma za izbor dimenzija matrica A i B za operacije zbrajanja, oduzimanja i množenja

Sljedeća forma imena „*Form2.cs*“ tiče se operacija koje zahtijevaju rad s dvije matrice, a to su: zbrajanje, oduzimanje i množenje.

Na slici 4.2. može se vidjeti izgled forme *Form2*.



Slika 4.2. Izgled forme *Form2*

Forma *Form2* sastoji se od sljedećih kontrola:

1. Label naziva „imeOperacije“ koji prikazuje naziv operacije odabrane iz početne forme *Form1* trenutno postavljen na zadani tekst „Ime operacije“.
2. Label koji prikazuje tekst „Unesite dimenzije matrica:“.
3. Label koji prikazuje tekst „Dimenzije matrice A:“.
4. Label koji prikazuje tekst „Dimenzije matrice B:“.
5. Label koji prikazuje tekst „Napomena: maksimalna dimenzija matrice je 8 x 8“.
6. Dva jednaka label-a koja prikazuju tekst „x“.

7. Button imena „potvrdiGumb“ koji predstavlja gumb za potvrdu unosa dimenzija matrica i prelazak na sljedeću formu *Form3*.
8. Button imena „povratakGumb“ koji predstavlja gumb za povratak na početnu formu *Form1*.
9. Dva TextBox-a imena „textBox1“ i „textBox2“ koji predstavljaju dimenzije matrice A.
10. Dva TextBox-a imena „textBox3“ i „textBox4“ koji predstavljaju dimenzije matrice B.

Sljedeći programski kod 4.4. sadrži četiri statičke *int* varijable n1, m1, n2, m2 koje predstavljaju dimenzije matrica pomoću kojih će se provjeravati je li korisnik ispravno unio dimenzije tih matrica. Također, unutar konstruktora klase *Form2* inicijalizirane su komponente forme te je tekst label-a „imeOperacije“ postavljen na tekst statičke *string* varijable „operacija“ iz početne forme *Form1* kako bi samo ime operacije pri vrhu forme bilo ispravno prikazano u odnosu na ono što je korisnik u početnoj formi izabrao. Nadalje, metoda „obrisiSve“ čisti sve TextBox elemente na način da briše sve što je zapisano u njima.

```
public static int n1, m1, n2, m2;

public Form2()
{
    InitializeComponent();
    imeOperacije.Text = Form1.operacija.ToUpper();
}

private void obrisiSve()
{
    textBox1.Clear();
    textBox2.Clear();
    textBox3.Clear();
    textBox4.Clear();
}
```

Programski kod 4.4. Četiri statičke *int* varijable, konstruktor i metoda „obrisiSve“

Sljedeća metoda prikazana u programskom kodu 4.5. predstavlja događaj koji se izvršava kada korisnik klikne na gumb imena „povratakGumb“.

```
private void povratakGumb_Click(object sender, EventArgs e)
{
    Hide();
    new Form1().ShowDialog();
    Close();
}
```

Programski kod 4.5. Događaj na klik gumba „povratakGumb“

Događaj koji se izvršava predstavlja već spomenuto sakrivanje i zatvaranje trenutne *Form2* forme, a otvaranje početne forme *Form1*.

Programski kod 4.6. predstavlja događaj koji se izvršava kada korisnik klikne na gumb imena „potvrdiGumb“.

```
private void potvrdiGumb_Click(object sender, EventArgs e)
{
    try
    {
        n1 = Convert.ToInt32(textBox1.Text);
        m1 = Convert.ToInt32(textBox2.Text);
        n2 = Convert.ToInt32(textBox3.Text);
        m2 = Convert.ToInt32(textBox4.Text);

        if (!(n1 >= 1 && n1 <= 8 && m1 >= 1 && m1 <= 8 &&
              n2 >= 1 && n2 <= 8 && m2 >= 1 && m2 <= 8))
        {
            MessageBox.Show("Neispravan unos dimenzija matrica!", "Pogreška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            obrisiSve();
        }

        else if ((Form1.operacija == "Zbrajanje" ||
                  Form1.operacija == "Oduzimanje") && (n1 != n2 || m1 != m2))
        {
            MessageBox.Show("Matrice moraju biti istih dimenzija!", "Pogreška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            obrisiSve();
        }

        else if ((Form1.operacija == "Množenje") && (m1 != n2))
        {
            MessageBox.Show("Matrice moraju biti ulančane!\nObjašnjenje: broj stupaca
matrice A mora biti jednak broju redaka matrice B.", "Pogreška", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            obrisiSve();
        }

        else
        {
            Hide();
            new Form3().ShowDialog();
            Close();
        }
    }
    catch
    {
        MessageBox.Show("Neispravan unos dimenzija matrica!", "Pogreška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        obrisiSve();
    }
}
```

Programski kod 4.6. Događaj na klik gumba „potvrdiGumb“

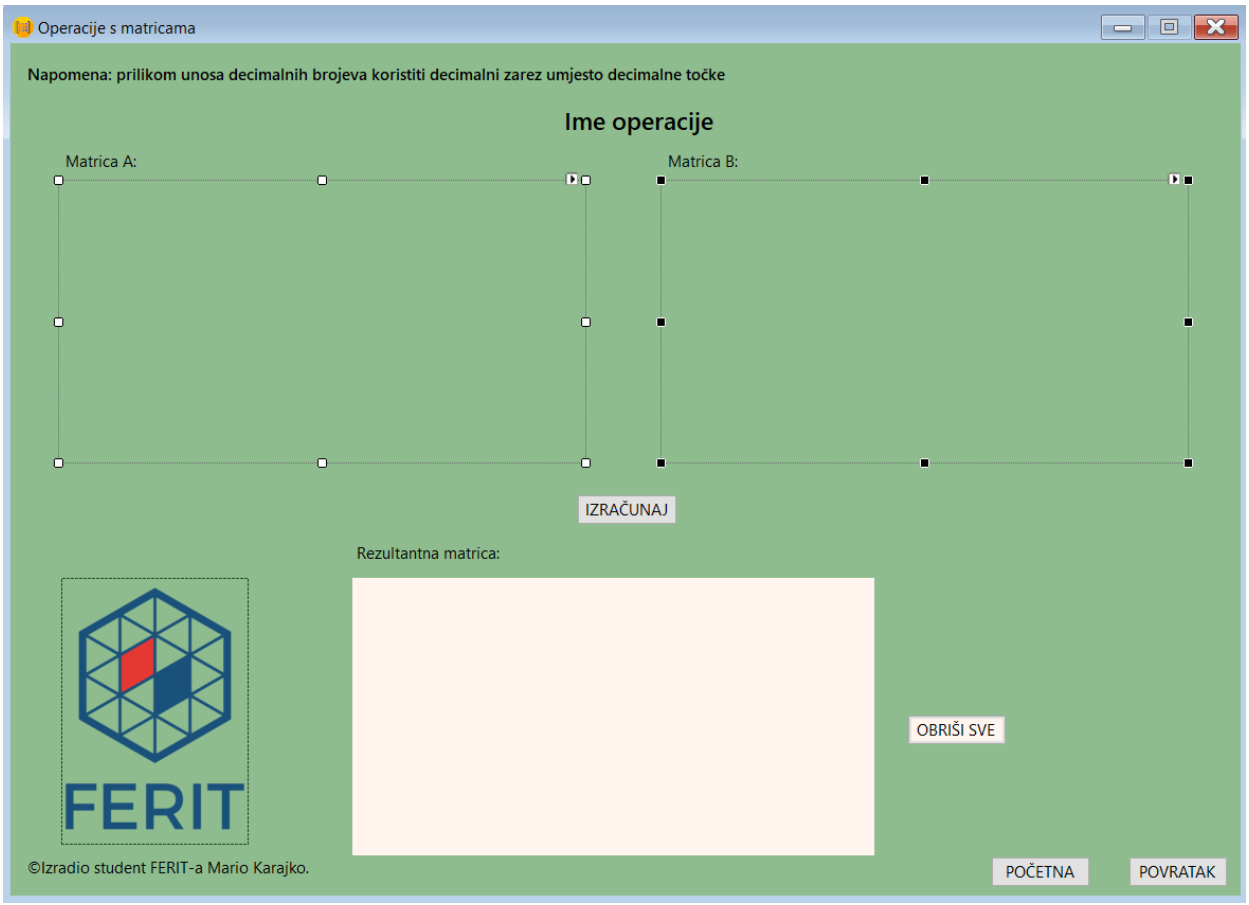
Ovaj je događaj smješten unutar *try-catch* bloka u slučaju ako korisnik umjesto ispravnog unosa dimenzija matrica unese nešto što ne može biti dimenzija, primjerice slovo „z“. U tom slučaju *catch* blok će biti izvršen te će se pojaviti prozorčić s porukom o pogrešci za neispravan unos dimenzija matrica. Također će i metoda „obrisiSve“ biti izvršena.

Try blok objašnjen je u nastavku. Prvo se u statičke *int* varijable spremaju brojevi koje je korisnik unio u TextBox-ove na način da se tekst, odnosno *string* iz TextBox-a pretvara u cijeli broj, tj. integer. Nakon toga, u slučaju da je neka od dimenzija matrica izvan intervala [1, 8], pojavit će se prozorčić s porukom o pogrešci za neispravan unos dimenzija matrica. U slučaju da je korisnik odabrao operaciju zbrajanja ili oduzimanja, a matice nisu istih dimenzija, pojavit će se prozorčić s porukom o pogrešci da matrice moraju biti jednakih dimenzija. U slučaju da je korisnik odabrao operaciju množenja, a matrice nisu ulančane, pojavit će se prozorčić s porukom o pogrešci da matrice moraju biti ulančane. U svim ovim *if* i *else if* uvjetima također će biti pozvana „obrisiSve“ metoda. Ukoliko niti jedan od prethodnih uvjeta nije bio zadovoljen, trenutna forma *Form2* bit će sakrivena i zatvorena, a sljedeća forma *Form3* otvorena.

4.3. Forma za izračun rezultantne matrice za operacije zbrajanja, oduzimanja i množenja

Sljedeća forma imena „*Form3.cs*“ odnosi se na upisivanje članova matrica A i B te na izračun rezultantne matrice ovisno o tome koju je operaciju korisnik u početnoj formi *Form1* odabrao.

Na slici 4.3. može se vidjeti izgled forme *Form3*.



Slika 4.3. Izgled forme *Form3*

Forma *Form3* sastoji se od sljedećih kontrola:

1. Label koji prikazuje tekst „Napomena: prilikom unosa decimalnih brojeva koristiti decimalni zarez umjesto decimalne točke“.
2. Label naziva „imeOperacije“ koji prikazuje naziv operacije odabrane iz početne forme *Form1* trenutno postavljen na zadani tekst „Ime operacije“.
3. Tri label-a koji prikazuju tekstove: „Matrica A:“, „Matrica B:“ te „Rezultantna matrica:“.
4. Button imena „izracunajGumb“ koji predstavlja gumb za izvršavanje odabrane operacije.
5. Button imena „obrisiGumb“ koji predstavlja gumb za brisanje rezultantne matrice i postavljanje matrica A i B na početno stanje kada su popunjene nulama.
6. Button imena „pocetnaGumb“ koji predstavlja gumb za povratak na početnu formu *Form1*.
7. Button imena „povratakGumb“ koji predstavlja gumb za povratak na prethodnu formu *Form2*.
8. DataGridView imena „dataGridView1“ koji predstavlja matricu A u „matričnom“ ili tabličnom obliku. Pozadinska boja ove kontrole postavljena je na „DarkSeaGreen“ boju kao što je i boja same forme.

9. DataGridView imena „dataGridView2“ koji predstavlja matricu B u „matričnom“ ili tabličnom obliku. Pozadinska boja ove kontrole također je postavljena na „DarkSeaGreen“ boju kao što je i boja same forme.
10. DataGridView imena „dataGridView3“ koji predstavlja rezultantnu matricu u „matričnom“ ili tabličnom obliku.

Sljedeći programski kod 4.7. sadrži konstruktor klase *Form3* te događaje koji se izvršavaju kada korisnik klikne na neki od gumbova „obrisiGumb“, „povratakGumb“ ili „pocetnaGumb“.

```
public Form3()
{
    InitializeComponent();
    imeOperacije.Text = Form1.operacija.ToUpper();
    popuniMatriceNulama();
}

private void obrisiGumb_Click(object sender, EventArgs e)
{
    dataGridView3.Columns.Clear();
    popuniMatriceNulama();
}

private void povratakGumb_Click(object sender, EventArgs e)
{
    Hide();
    new Form2().ShowDialog();
    Close();
}

private void pocetnaGumb_Click(object sender, EventArgs e)
{
    Hide();
    new Form1().ShowDialog();
    Close();
}
```

Programski kod 4.7. Konstruktor te događaji na klik gumbova „obrisiGumb“, „povratakGumb“ i „pocetnaGumb“

Unutar programskog koda 4.7. može se primijetiti nekoliko stavki. Kao prvo, u konstruktoru klase *Form3* inicijaliziraju se komponente forme, tekst label-a „imeOperacije“ postavlja se na tekst statičke *string* varijable „operacija“ iz početne forme *Form1*, ali s velikim slovima te se poziva metoda *popuniMatriceNulama* koja će kasnije biti detaljno objašnjena. Nadalje, događaj koji se izvršava kada korisnik klikne na gumb „obrisiGumb“ briše, odnosno „čisti“ kompletnu rezultantnu matricu, tj. cijeli *dataGridView3* element te također popunjava matrice A i B s nulama. Sljedeći događaj „povratakGumb_Click“ sakriva i zatvara trenutnu *Form3* formu i otvara prethodnu *Form2* formu. Posljednji događaj „pocetnaGumb_Click“ predstavlja sakrivanje i zatvaranje trenutne *Form3* forme i otvaranje početne *Form1* forme.

Nadalje, potrebno je objasniti metode *popuniMatriceNulama*, *Zbrajanje*, *Oduzimanje*, *Mnozenje* te događaj „izracunajGumb_Click“.

Metoda *popuniMatriceNulama* prikazana je unutar programskog koda 4.8..

```
private void popuniMatriceNulama()
{
    int n1, m1, n2, m2;
    n1 = Form2.n1;
    m1 = Form2.m1;
    n2 = Form2.n2;
    m2 = Form2.m2;
    decimal[,] matrixA = new decimal[n1, m1];
    decimal[,] matrixB = new decimal[n2, m2];
    DataTable dt1 = new DataTable();
    DataTable dt2 = new DataTable();
    for (int j = 0; j < matrixA.GetLength(1); j++)
    {
        dt1.Columns.Add("A" + (j + 1).ToString());
    }

    for (int j = 0; j < matrixB.GetLength(1); j++)
    {
        dt2.Columns.Add("B" + (j + 1).ToString());
    }

    DataRow dr1, dr2;
    for (int i = 0; i < matrixA.GetLength(0); i++)
    {
        dr1 = dt1.NewRow();
        for (int j = 0; j < matrixA.GetLength(1); j++)
        {
            dr1[j] = matrixA[i, j];
        }
        dt1.Rows.Add(dr1);
    }

    for (int i = 0; i < matrixB.GetLength(0); i++)
    {
        dr2 = dt2.NewRow();
        for (int j = 0; j < matrixB.GetLength(1); j++)
        {
            dr2[j] = matrixB[i, j];
        }
        dt2.Rows.Add(dr2);
    }
    dataGridView1.DataSource = dt1;
    foreach (DataGridViewColumn column in dataGridView1.Columns)
    {
        column.SortMode = DataGridViewColumnSortMode.NotSortable;
    }
    dataGridView2.DataSource = dt2;
    foreach (DataGridViewColumn column in dataGridView2.Columns)
    {
        column.SortMode = DataGridViewColumnSortMode.NotSortable;
    }
}
```

Programski kod 4.8. Metoda *popuniMatriceNulama*

Programski kod 4.8. bit će objašnjen slijedno. Za početak, definirane su četiri *int* varijable n1, m1, n2, m2 koje predstavljaju dimenzije matrica A i B. Potom slijedi njihova deklaracija pomoću istoimenih varijabli koje se prenose iz forme *Form2*. Nakon toga definirane su matrice A i B tipa *decimal* te dva objekta *dt1* i *dt2* tipa *DataTable* pomoću kojih će sadržaj matrica A i B biti prikazan. U sljedeće dvije *for* petlje dodani su nazivi stupaca svake matrice na način da je primjerice prvi stupac matrice A označen s „A1“, drugi stupac s „A2“ itd. Zatim slijedi popunjavanje objekta *dt1* i to na način da se pomoću prve *for* petlje *dt1* proširuje red po red objektom *dr1* tipa *DataRow*, a pomoću druge *for* petlje sami se element na *j* poziciji postavlja na vrijednost *matrixA*[*i, j*] matrice A koja je jednaka nuli jer se prilikom definiranja samog objekta *matrixA* tipa koji je tipa *decimal* sve njegove vrijednosti zadano postavljaju na nulu. Kada druga *for* petlja „prođe“ kroz sve stupce, taj se red dodaje u *dt1* objekt. Analogno se popunjava i *dt2* objekt. Naposljetku se kao izvor podataka (engl. *DataSource*) za *dataGridView1* objekt postavlja *DataTable* objekt *dt1*, a za *dataGridView2* objekt *dt2*. Sami stupci svakog DataGridView objekta fiksirani su tako da im je onemogućeno sortiranje kako se ne bi pomiješao sam razmještaj elemenata stupaca tablice.

Metoda *Zbrajanje* prikazana je unutar programskog koda 4.9..

```
private void Zbrajanje(decimal[,] matrixA, decimal[,] matrixB)
{
    decimal[,] matrixC = new decimal[matrixA.GetLength(0), matrixA.GetLength(1)];
    for (int i = 0; i < matrixC.GetLength(0); i++)
    {
        for (int j = 0; j < matrixC.GetLength(1); j++)
        {
            matrixC[i, j] = matrixA[i, j] + matrixB[i, j];
        }
    }
    DataTable dt = new DataTable();
    for (int j = 0; j < matrixC.GetLength(1); j++)
    {
        dt.Columns.Add("C" + (j + 1).ToString());
    }
    DataRow dr;
    for (int i = 0; i < matrixC.GetLength(0); i++)
    {
        dr = dt.NewRow();
        for (int j = 0; j < matrixC.GetLength(1); j++)
        {
            dr[j] = Math.Round((decimal)matrixC[i, j], 5);
        }
        dt.Rows.Add(dr);
    }
    dataGridView3.DataSource = dt;
    foreach (DataGridViewColumn column in dataGridView3.Columns)
    {
        column.SortMode = DataGridViewColumnSortMode.NotSortable;
    }
}
```

Programski kod 4.9. Metoda za zbrajanje dviju matrica

Programski kod 4.9. za zbrajanje matrica objašnjen je u nastavku. Metoda *Zbrajanje* sastoji se od dva parametra: matrice A i matrice B koje su tipa *decimal*. Na početku metode definirana je matrica C tipa *decimal* koja je dimenzija matrice A. Naime, matrica C može biti i dimenzija matrice B budući da dimenzije matrice A i B moraju biti jednake budući da se radi o operaciji zbrajanja kod koje matrice moraju biti istog tipa, tj. istih dimenzija. Nakon definicije matrice C, u sljedeće dvije *for* petlje svakom elementu matrice C na poziciji [*i, j*] pridružena je vrijednost jednaka zbroju matrica A i B na istoj toj poziciji. Tako je matrica C popunjena odgovarajućim vrijednostima zbroja matrica A i B. Poslije toga dolazi kod koji je objašnjen već u programskom kodu 4.8. samo što se u ovome slučaju radi o rezultantnoj matrici C čiji se elementi zaokružuju na pet decimala.

Za operaciju oduzimanja princip je isti kao kod metode *Zbrajanje* samo što se unutar druge *for* petlje nakon definicije matrice C element matrice A na poziciji [*i, j*] oduzima od elementa matrice B na istoj toj poziciji. Programski kod 4.10. za metodu *Oduzimanje* dan je u nastavku:

```
private void Oduzimanje(decimal[,] matrixA, decimal[,] matrixB)
{
    decimal[,] matrixC = new decimal[matrixA.GetLength(0), matrixA.GetLength(1)];

    for (int i = 0; i < matrixC.GetLength(0); i++)
    {
        for (int j = 0; j < matrixC.GetLength(1); j++)
        {
            matrixC[i, j] = matrixA[i, j] - matrixB[i, j];
        }
    }

    DataTable dt = new DataTable();
    for (int j = 0; j < matrixC.GetLength(1); j++)
    {
        dt.Columns.Add("C" + (j + 1).ToString());
    }

    DataRow dr;
    for (int i = 0; i < matrixC.GetLength(0); i++)
    {
        dr = dt.NewRow();
        for (int j = 0; j < matrixC.GetLength(1); j++)
        {
            dr[j] = Math.Round((decimal)matrixC[i, j], 5);
        }
        dt.Rows.Add(dr);
    }
    dataGridView3.DataSource = dt;
    foreach (DataGridViewColumn column in dataGridView3.Columns)
    {
        column.SortMode = DataGridViewColumnSortMode.NotSortable;
    }
}
```

Programski kod 4.10. Metoda za oduzimanje dviju matrica

Operacija množenja malo je složenija od operacija zbrajanja i oduzimanja. Programski kod 4.11. za operaciju množenja dan je u nastavku:

```
private void Mnozenje(decimal[,] matrixA, decimal[,] matrixB)
{
    decimal[,] matrixC = new decimal[matrixA.GetLength(0), matrixB.GetLength(1)];

    for (int i = 0; i < matrixC.GetLength(0); i++)
    {
        for (int j = 0; j < matrixC.GetLength(1); j++)
        {
            matrixC[i, j] = 0.0m;
            for (int k = 0; k < matrixA.GetLength(1); k++)
            {
                matrixC[i, j] += matrixA[i, k] * matrixB[k, j];
            }
        }
    }

    DataTable dt = new DataTable();
    for (int j = 0; j < matrixC.GetLength(1); j++)
    {
        dt.Columns.Add("C" + (j + 1).ToString());
    }

    DataRow dr;
    for (int i = 0; i < matrixC.GetLength(0); i++)
    {
        dr = dt.NewRow();
        for (int j = 0; j < matrixC.GetLength(1); j++)
        {
            dr[j] = Math.Round((decimal)matrixC[i, j], 4);
        }
        dt.Rows.Add(dr);
    }
    dataGridView3.DataSource = dt;
    foreach (DataGridViewColumn column in dataGridView3.Columns)
    {
        column.SortMode = DataGridViewColumnSortMode.NotSortable;
    }
}
```

Programski kod 4.11. Metoda za množenje dviju matrica

Objašnjenje za programski kod 4.11. je u nastavku. Potrebno je prvo napomenuti da je uvjet za množenje matrica taj da su matrice ulančane, tj. broj stupaca prve matrice mora biti jednak broju redaka druge matrice. Zato se na početku definira matrica C koja ima broj redaka jednak broju redaka matrice A te broj stupaca jednak broju stupaca matrice B. Nadalje, da bi se dobila vrijednost elementa rezultantne matrice C na poziciji [*i, j*] potrebne su tri *for* petlje. Prve dvije petlje jednake su kao kod operacija zbrajanja i oduzimanja, a još su dodani inicijalizacija elementa matrice C na poziciji [*i, j*] na vrijednost nula te treća *for* petlja kojom će se dodijeliti vrijednost koja će biti jednaka zbroju elemenata matrice A na poziciji [*i, k*] i elemenata matrice B na poziciji [*k, j*]. Pri tome indeks *k* kreće od pozicije prvog elementa matrice u trenutnom retku i stupcu i ide sve do

pozicije zadnjeg elementa kojemu je pozicija jednaka vrijednosti zajedničkoj za matrice A i B, a to je broj stupaca matrice A ili broj redaka matrice B. Na taj način je omogućeno *tzv.* „šetanje“ po stupcima matrice A i redcima matrice B. Nakon izračuna vrijednosti elementa matrice C na poziciji [*i, j*] koja se zaokružuje na četiri decimale, vrijednost sljedećeg elementa mora se postaviti na nulu kako u toj varijabli ne bi ostala sačuvana vrijednost prethodnog elementa.

Još je preostalo objasniti događaj koji se izvodi klikom na „izracunajGumb“. Programski kod 4.12. za taj događaj dan je u nastavku:

```
private void izracunajGumb_Click(object sender, EventArgs e)
{
    int n1, m1, n2, m2;
    n1 = Form2.n1;
    m1 = Form2.m1;
    n2 = Form2.n2;
    m2 = Form2.m2;
    decimal[,] matrixA = new decimal[n1, m1];
    decimal[,] matrixB = new decimal[n2, m2];
    try
    {
        for (int i = 0; i < matrixA.GetLength(0); i++)
        {
            for (int j = 0; j < matrixA.GetLength(1); j++)
            {
                matrixA[i, j] =
Convert.ToDecimal(dataGridView1.Rows[i].Cells[j].Value);
            }
        }

        for (int i = 0; i < matrixB.GetLength(0); i++)
        {
            for (int j = 0; j < matrixB.GetLength(1); j++)
            {
                matrixB[i, j] =
Convert.ToDecimal(dataGridView2.Rows[i].Cells[j].Value);
            }
        }

        if (Form1.operacija == "Zbrajanje") {
            Zbrajanje(matrixA, matrixB);
        } else if (Form1.operacija == "Oduzimanje") {
            Oduzimanje(matrixA, matrixB);
        } else if (Form1.operacija == "Množenje") {
            Mnozenje(matrixA, matrixB);
        }
    }
    catch
    {
        MessageBox.Show("Neispravan unos člana/ova matrice!", "Pogreška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        obrisiGumb_Click(sender, e);
    }
}
```

Programski kod 4.12. Događaj na klik gumba „izracunajGumb“

Objašnjene programskog koda 4.12. dano je u nastavku. Na početku metode u varijable n1, m1, n2, m2 preuzete su vrijednosti dimenzija matrica A i B iz forme *Form2*. Potom su definirane matrice A i B tipa *decimal* tih dimenzija. Zatim slijedi kod koji je stavljen u *try-catch* blok u slučaju ako korisnik unese neispravan unos jednog ili više članova matrice A ili matrice B. Zatim se pomoću dvije *for* petlje u svaki element matrice A upisuje vrijednost *dataGridView1* objekta na toj poziciji pri čemu se ta vrijednost mora konvertirati u tip *decimal* budući da je sama matrica A tog tipa. Analogno se radi i za matricu B. Na kraju se pomoću *if* i *else if* uvjeta provjerava koja je operacija iz početne forme *Form1* odabrana te se na temelju toga izvodi neka od metoda „*Zbrajanje*“, „*Oduzimanje*“ ili „*Mnozenje*“ s argumentima *matrixA* i *matrixB*. U slučaju da je u nekom trenutku korisnik u matricu A i/ili matricu B unio neku vrijednost koja se ne može konvertirati u tip *decimal* izvršava se *catch* blok kojim se korisniku prikazuje prozorčić s porukom o pogrešci za neispravan unos člana/ova matrice te se izvršava događaj „obrisiGumb_Click“ kojim se „čiste“ sve matrice na početno stanje.

4.4. Forma za izbor dimenzija matrice i skalara za operaciju množenja sa skalarom

Forma „*Form4.cs*“ koristi se za izbor dimenzija matrice i skalara koji će se koristiti u narednoj formi *Form5*.

Na slici 4.4. može se vidjeti izgled forme *Form4*.

Slika 4.4. Izgled forme *Form4*

Forma *Form4* sastoji se od sljedećih kontrola:

1. Label koji prikazuje tekst „MNOŽENJE SA SKALAROM“.
2. Label koji prikazuje tekst „Unesite dimenzije matrice i skalar:“.
3. Label koji prikazuje tekst „Dimenzije matrice:“.
4. Label koji prikazuje tekst „Skalar:“.
5. Label koji prikazuje tekst „Napomena: maksimalna dimenzija matrice je 8 x 8“.
6. Label koji prikazuje tekst „Napomena: skalar se unosi pomoću decimalnog zareza, a ne pomoću decimalne točke“.
7. Label koji prikazuje tekst „x“.
8. Button imena „potvrdiGumb“ koji predstavlja gumb za potvrdu unosa dimenzija matrice i skalara te prelazak na sljedeću formu *Form5*.
9. Button imena „povratakGumb“ koji predstavlja gumb za povratak na početnu formu *Form1*.
10. Dva TextBox-a imena „textBox1“ i „textBox2“ koji predstavljaju dimenzije matrice.
11. TextBox-a imena „skalarTextBox“ koji predstavlja skalar s kojim će se matrica množiti.

Sljedeći programski kod 4.13. sadrži dvije statičke *int* varijable n i m koje predstavljaju dimenzije matrice te statičku *decimal* varijablu skalar. Pomoću njih će se provjeravati je li korisnik ispravno unio dimenzije te matrice i skalar. Također, unutar konstruktora klase *Form4* inicijalizirane su komponente forme. Nadalje, metoda „obrisiSve“ čisti sve TextBox elemente (textBox1, textBox2 i skalarTextBox) na način da briše sve što je zapisano u njima.

```
public static decimal skalar;
public static int n, m;
public Form4()
{
    InitializeComponent();
}

private void obrisiSve()
{
    textBox1.Clear();
    textBox2.Clear();
    skalarTextBox.Clear();
}
```

Programski kod 4.13. Dvije statičke *int* varijable, statička *decimal* varijabla, konstruktor i metoda „obrisiSve“

Sljedeća metoda prikazana u programskom kodu 4.14. predstavlja događaj koji se izvršava kada korisnik klikne na gumb imena „povratakGumb“.

```
private void povratakGumb_Click(object sender, EventArgs e)
{
    Hide();
    new Form1().ShowDialog();
    Close();
}
```

Programski kod 4.14. Događaj na klik gumba „povratakGumb“

Događaj koji se izvršava predstavlja već spomenuto sakrivanje i zatvaranje trenutne *Form4* forme, a otvaranje početne forme *Form1*.

Programski kod 4.15. predstavlja događaj koji se izvršava kada korisnik klikne na gumb imena „potvrdiGumb“.

```
private void potvrdiGumb_Click(object sender, EventArgs e)
{
    try
    {
        n = Convert.ToInt32(textBox1.Text);
        m = Convert.ToInt32(textBox2.Text);
        skalar = Convert.ToDecimal(skalarTextBox.Text);
        if (!(n >= 1 && n <= 8 && m >= 1 && m <= 8))
        {
            MessageBox.Show("Neispravan unos dimenzija matrice!", "Pogreška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            obrisiSve();
        }
        else
        {
            Hide();
            new Form5().ShowDialog();
            Close();
        }
    }
    catch
    {
        MessageBox.Show("Neispravan unos dimenzije matrice i/ili skalara!",
"Pogreška", MessageBoxButtons.OK, MessageBoxIcon.Error);
        obrisiSve();
    }
}
```

Programski kod 4.15. Događaj na klik gumba „potvrdiGumb“

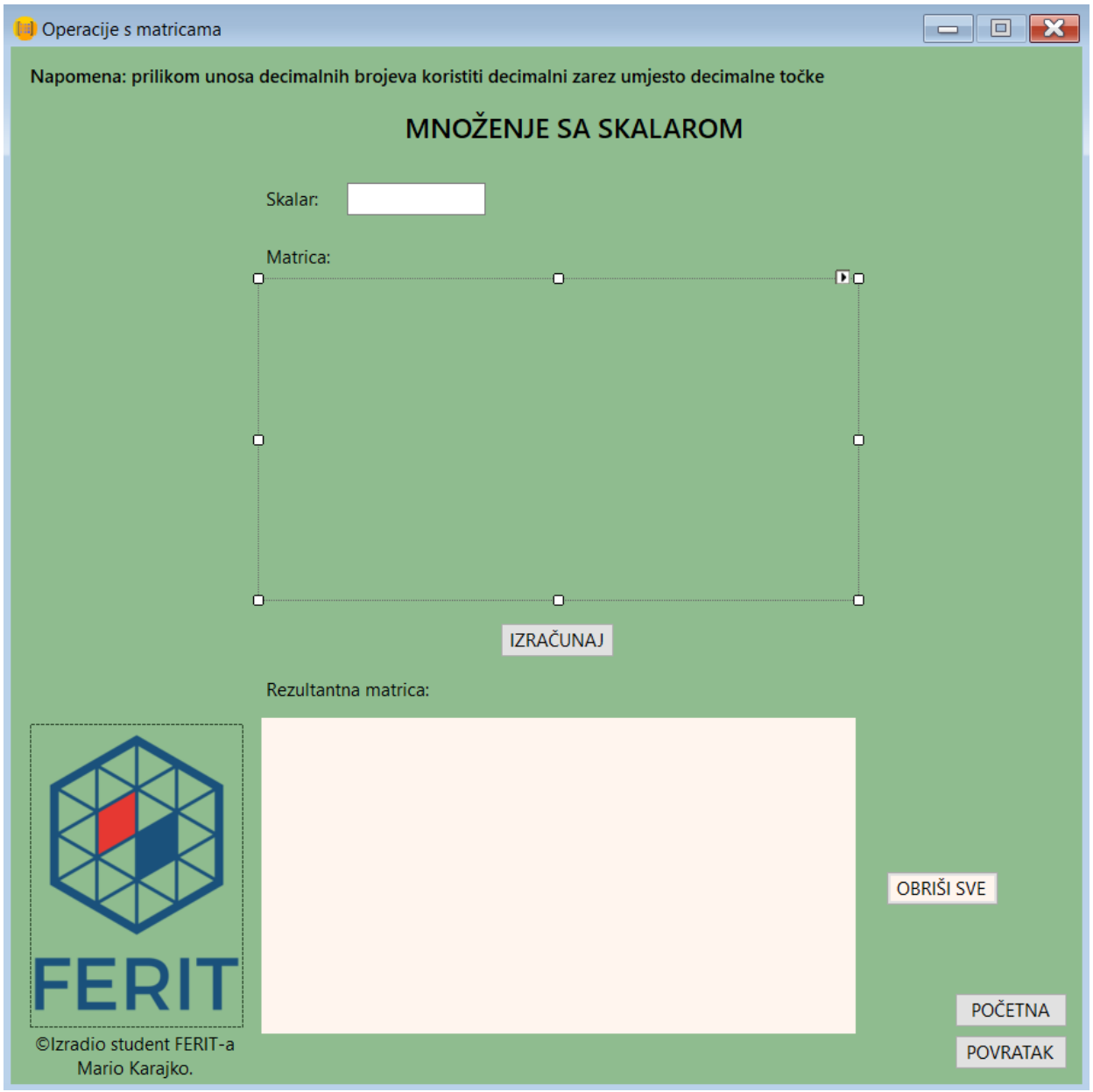
Ovaj je događaj smješten unutar *try-catch* bloka u slučaju ako korisnik umjesto ispravnog unosa dimenzija matrica i/ili skalara unese nešto što ne može biti dimenzija i/ili skalar, primjerice neko slovo ili tekst. U tom slučaju *catch* blok će biti izvršen te će se pojaviti prozorčić s porukom o pogrešci za neispravan unos dimenzija matrice i/ili skalara. Također će i metoda „obrisiSve“ biti izvršena.

Try blok objašnjen je u nastavku. Prvo se u statičke *int* varijable n i m te statičku *decimal* varijablu skalar spremaju brojevi koje je korisnik unio u TextBox-ove na način da se tekst iz TextBox-a pretvori u cijeli broj odnosno u decimalni broj u slučaju skalara. Nakon toga, u slučaju da je neka od dimenzija matrice izvan intervala [1, 8], pojavit će se prozorčić s porukom o pogrešci za neispravan unos dimenzija matrice i bit će pozvana „obrisiSve“ metoda. Ukoliko prethodni uvjet nije bio zadovoljen, trenutna forma *Form4* bit će sakrivena i zatvorena, a sljedeća forma *Form5* otvorena.

4.5. Forma za izračun rezultantne matrice za operaciju množenja sa skalarom

Zadnja forma imena „*Form5.cs*“ odnosi se na upisivanje članova matrice i upis ili izmjenu skalara te na izračun rezultantne matrice.

Na slici 4.5. može se vidjeti izgled forme *Form5*.



Slika 4.5. Izgled forme *Form5*

Forma *Form5* sastoji se od sljedećih kontrola:

1. Label koji prikazuje tekst „Napomena: prilikom unosa decimalnih brojeva koristiti decimalni zarez umjesto decimalne točke“.
2. Label koji prikazuje tekst „MNOŽENJE SA SKALAROM“.
3. Tri label-a koji prikazuju tekstove: „Skalar:“, „Matrica:“ te „Rezultantna matrica:“.
4. TextBox-a imena „skalarTextBox1“ koji predstavlja skalar s kojim će se matrica množiti.
5. Button imena „izracunajGumb“ koji predstavlja gumb za izvršavanje operacije množenja sa skalarom.

6. Button imena „obrisiGumb“ koji predstavlja gumb za brisanje rezultantne matrice i postavljanje matrice na početno stanje kada je popunjena nulama te brisanje TextBox-a sa skalarom.
7. Button imena „pocetnaGumb“ koji predstavlja gumb za povratak na početnu formu *Form1*.
8. Button imena „povratakGumb“ koji predstavlja gumb za povratak na prethodnu formu *Form4*.
9. DataGridView imena „dataGridView1“ koji predstavlja matricu u „matričnom“ ili tabličnom obliku. Pozadinska boja ove kontrole postavljena je na „DarkSeaGreen“ boju kao što je i boja same forme.
10. DataGridView imena „dataGridView3“ koji predstavlja rezultantnu matricu u „matričnom“ ili tabličnom obliku.

Sljedeći programski kod 4.16. sadrži konstruktor klase *Form5* te događaje koji se izvršavaju kada korisnik klikne na neki od gumbova „obrisiGumb“, „povratakGumb“ ili „pocetnaGumb“.

```
public Form5()
{
    InitializeComponent();
    skalarTextBox1.Text = Form4.skalar.ToString();
    popuniMatricuNulama();
}

private void obrisiGumb_Click(object sender, EventArgs e)
{
    skalarTextBox1.Clear();
    popuniMatricuNulama();
    dataGridView3.Columns.Clear();
}

private void povratakGumb_Click(object sender, EventArgs e)
{
    Hide();
    new Form4().ShowDialog();
    Close();
}

private void pocetnaGumb_Click(object sender, EventArgs e)
{
    Hide();
    new Form1().ShowDialog();
    Close();
}
```

Programski kod 4.16. Konstruktor te događaji na klik gumbova „obrisiGumb“, „povratakGumb“ i „pocetnaGumb“

Unutar programskog koda 4.16. može se primijetiti nekoliko stavki. Kao prvo, u konstruktoru klase *Form5* inicijaliziraju se komponente forme, tekst „skalarTextBox1“ TextBox-a postavlja se

na vrijednost statičke *decimal* varijable skalar prenesene iz prethodne forme *Form4*, ali pretvorenu u tip *string* te se poziva metoda *popuniMatricuNulama*. Nadalje, događaj koji se izvršava kada korisnik klikne na gumb „obrisiGumb“ popunjava matricu s nulama, „čisti“ „skalarTextBox1“ te briše, odnosno „čisti“ kompletnu rezultantnu matricu, tj. cijeli *dataGridView3* element. Sljedeći događaj „povratakGumb_Click“ sakriva i zatvara trenutnu *Form5* formu i otvara prethodnu *Form4* formu. Posljednji događaj „pocetnaGumb_Click“ predstavlja sakrivanje i zatvaranje trenutne *Form5* forme i otvaranje početne *Form1* forme.

Metoda *popuniMatricuNulama* radi na jednakom principu kao i metoda *popuniMatriceNulama* iz potpoglavlja 4.3. (programski kod 4.8.) samo što je u ovome slučaju prisutna samo jedna matrica, a ne dvije.

Programski kod 4.17. za metodu *popuniMatricuNulama* dan je u nastavku:

```
private void popuniMatricuNulama()
{
    int n, m;
    n = Form4.n;
    m = Form4.m;
    decimal[,] matrix = new decimal[n, m];
    DataTable dt = new DataTable();
    for (int j = 0; j < matrix.GetLength(1); j++)
    {
        dt.Columns.Add("A" + (j + 1).ToString());
    }
    DataRow dr;
    for (int i = 0; i < matrix.GetLength(0); i++)
    {
        dr = dt.NewRow();
        for (int j = 0; j < matrix.GetLength(1); j++)
        {
            dr[j] = matrix[i, j];
        }
        dt.Rows.Add(dr);
    }
    dataGridView1.DataSource = dt;
    foreach (DataGridViewColumn column in dataGridView1.Columns)
    {
        column.SortMode = DataGridViewColumnSortMode.NotSortable;
    }
}
```

Programski kod 4.17. Metoda *popuniMatricuNulama*

Metoda *MnozenjeSaSkalarom* prikazana je unutar programskog koda 4.18..

```
private void MnozenjeSaSkalarom(decimal skalar, decimal[,] matrix)
{
    decimal[,] matrixC = new decimal[matrix.GetLength(0), matrix.GetLength(1)];

    for (int i = 0; i < matrixC.GetLength(0); i++)
    {
        for (int j = 0; j < matrixC.GetLength(1); j++)
        {
            matrixC[i, j] = skalar * matrix[i, j];
        }
    }

    DataTable dt = new DataTable();
    for (int j = 0; j < matrixC.GetLength(1); j++)
    {
        dt.Columns.Add("C" + (j + 1).ToString());
    }

    DataRow dr;
    for (int i = 0; i < matrixC.GetLength(0); i++)
    {
        dr = dt.NewRow();
        for (int j = 0; j < matrixC.GetLength(1); j++)
        {
            dr[j] = Math.Round((decimal)matrixC[i, j], 5);
        }
        dt.Rows.Add(dr);
    }

    dataGridView3.DataSource = dt;
    foreach (DataGridViewColumn column in dataGridView3.Columns)
    {
        column.SortMode = DataGridViewColumnSortMode.NotSortable;
    }
}
```

Programski kod 4.18. Metoda za množenje matrice sa skalarom

Ovaj princip dodavanja redaka u *DataTable* objekt i postavljanje izvora podataka DataGridView elementa već je prethodno objašnjeno, a sama operacija množenja elementa matrice sa skalarom nije kompliciran. Jednostavno se pomoću dvije *for* petlje prolazi kroz svaki element, a vrijednost elementa na poziciji [*i, j*] množi se sa skalarom koji se predaje kao argument ove metode te se element nakraju zaokružuje na pet decimala.

Preostalo je još objasniti događaj prilikom klika na „izracunajGumb“.

Programski kod 4.19. za taj događaj dan je u nastavku.

```
private void izracunajGumb_Click(object sender, EventArgs e)
{
    try
    {
        int n, m;
        decimal skalar;
        n = Form4.n;
        m = Form4.m;
        skalar = Convert.ToDecimal(skalarTextBox1.Text);
        decimal[,] matrix = new decimal[n, m];
        for (int i = 0; i < matrix.GetLength(0); i++)
        {
            for (int j = 0; j < matrix.GetLength(1); j++)
            {
                matrix[i, j] =
Convert.ToDecimal(dataGridView1.Rows[i].Cells[j].Value);
            }
        }
        MnozenjeSaSkalarom(skalar, matrix);
    }
    catch
    {
        MessageBox.Show("Neispravan unos člana/ova matrice i/ili skalara!",
"Pogreška", MessageBoxButtons.OK, MessageBoxIcon.Error);
        obrisiGumb_Click(sender, e);
    }
}
```

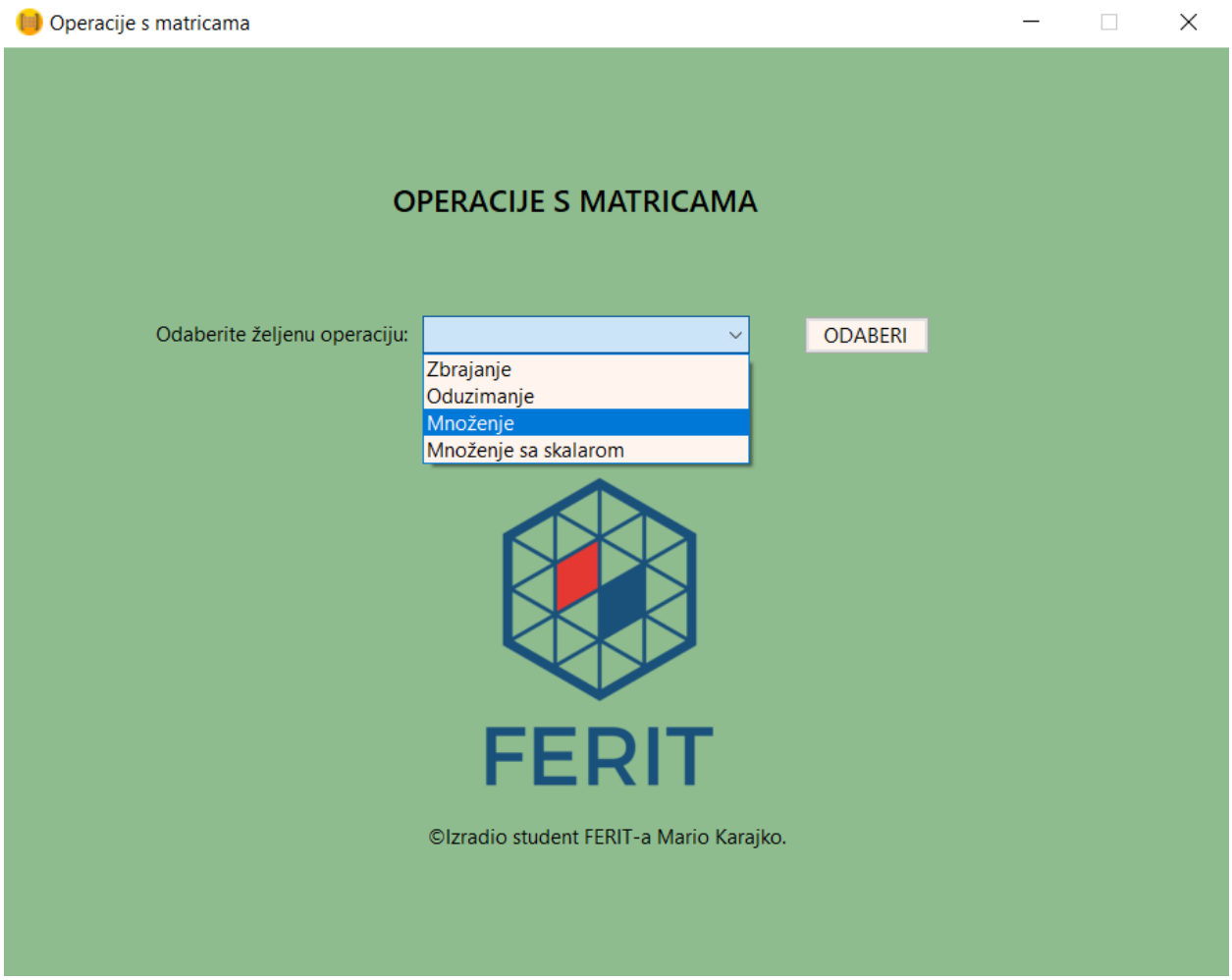
Programski kod 4.19. Događaj na klik gumba „izracunajGumb“

Objašnjene programskog koda 4.19. dano je u nastavku. Cjelokupan je kod stavljen unutar bloka *try-catch* u slučaju ako korisnik unese neispravan unos člana/ova matrice i/ili skalara. Na početku metode u varijable n i m preuzete su vrijednosti dimenzija matrice iz forme *Form4*, a u varijablu *skalar* spremljena je vrijednost iz TextBox-a „skalarTextBox1“, ali konvertirana u *decimal* tip podatka. Potom je definirana matrica tipa *decimal* dimenzija n i m. Zatim se pomoću dvije *for* petlje u svaki element matrice upisuje vrijednost *dataGridView1* objekta na toj poziciji pri čemu se ta vrijednost mora konvertirati u tip *decimal* budući da je sama matrica tog tipa. Na kraju se poziva sama metoda *MnozenjeSaSkalarom* kojoj se predaju parametri *skalar* i *matrix*. U slučaju da je u nekom trenutku korisnik u matricu ili u TextBox „skalarTextBox1“ unio neku vrijednost koja se ne može konvertirati u tip *decimal* izvršava se *catch* blok kojim se korisniku prikazuje prozorčić s porukom o pogrešci za neispravan unos člana/ova matrice i/ili skalara te se izvršava događaj „obrisiGumb_Click“ kojim se „čiste“ sve matrice na početno stanje i briše „skalarTextBox1“.

5. IZGLED APLIKACIJE

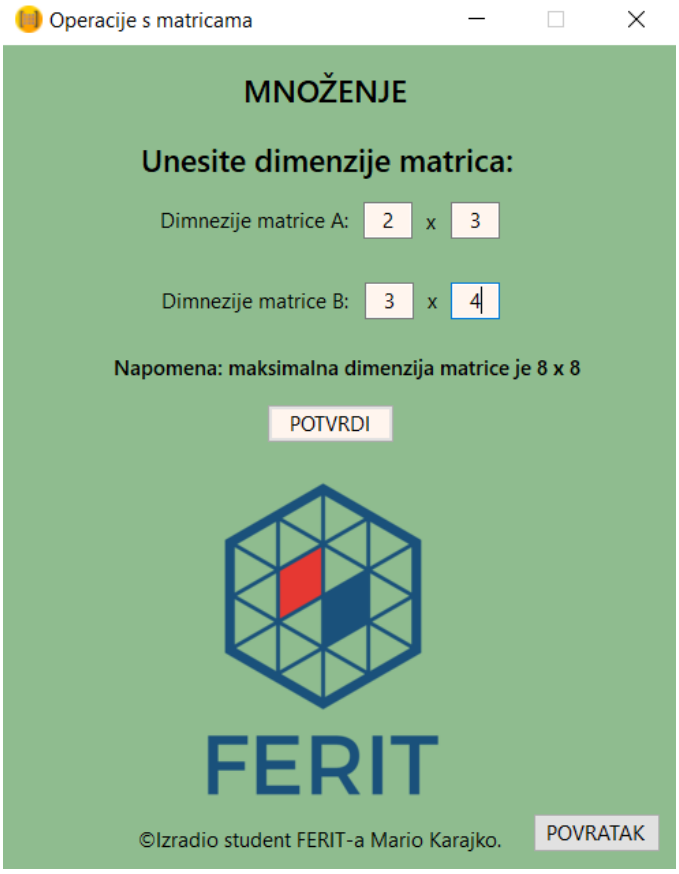
Konačni slijedni pregled kroz aplikaciju bit će prikazan u nastavku.

Za početak se u početnom prozoru u padajućem izborniku izabere npr. operacija množenja i klikne na gumb „ODABERI“. Prikaz se može vidjeti na slici 5.1.



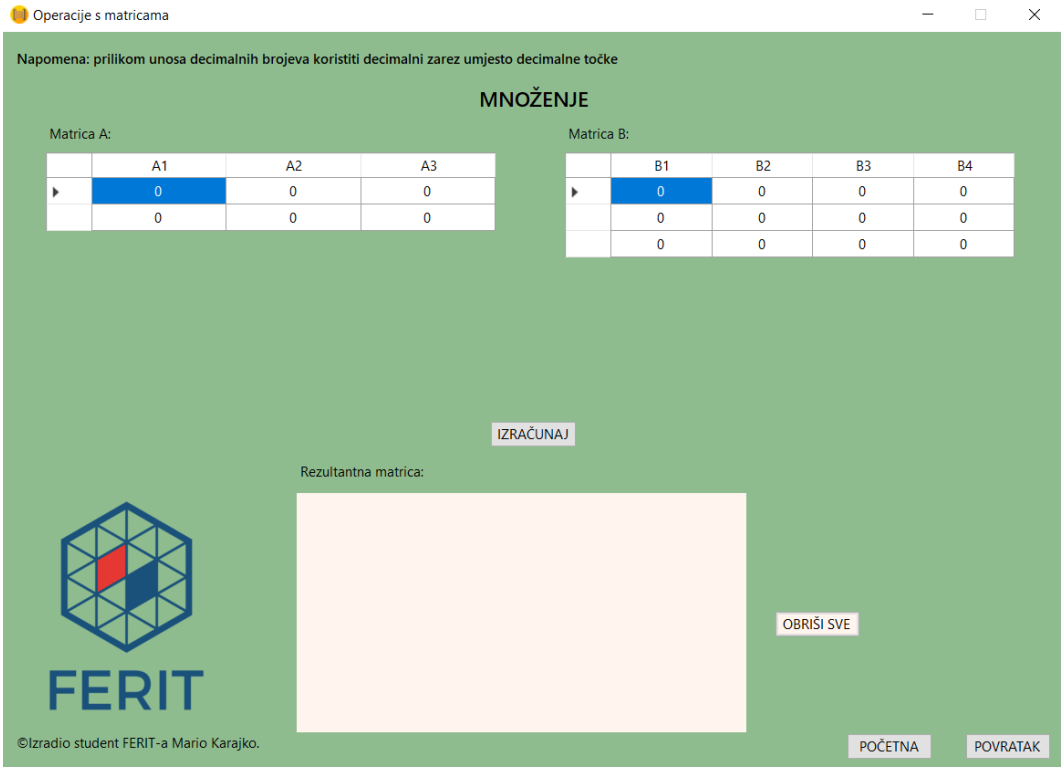
Slika 5.1. Odabir operacije množenja

U sljedećem prozoru ćemo upisati dimenzije matrica A i B tako da matrice budu ulančane. Izgled trenutnog prozora prikazan je na slici 5.2.



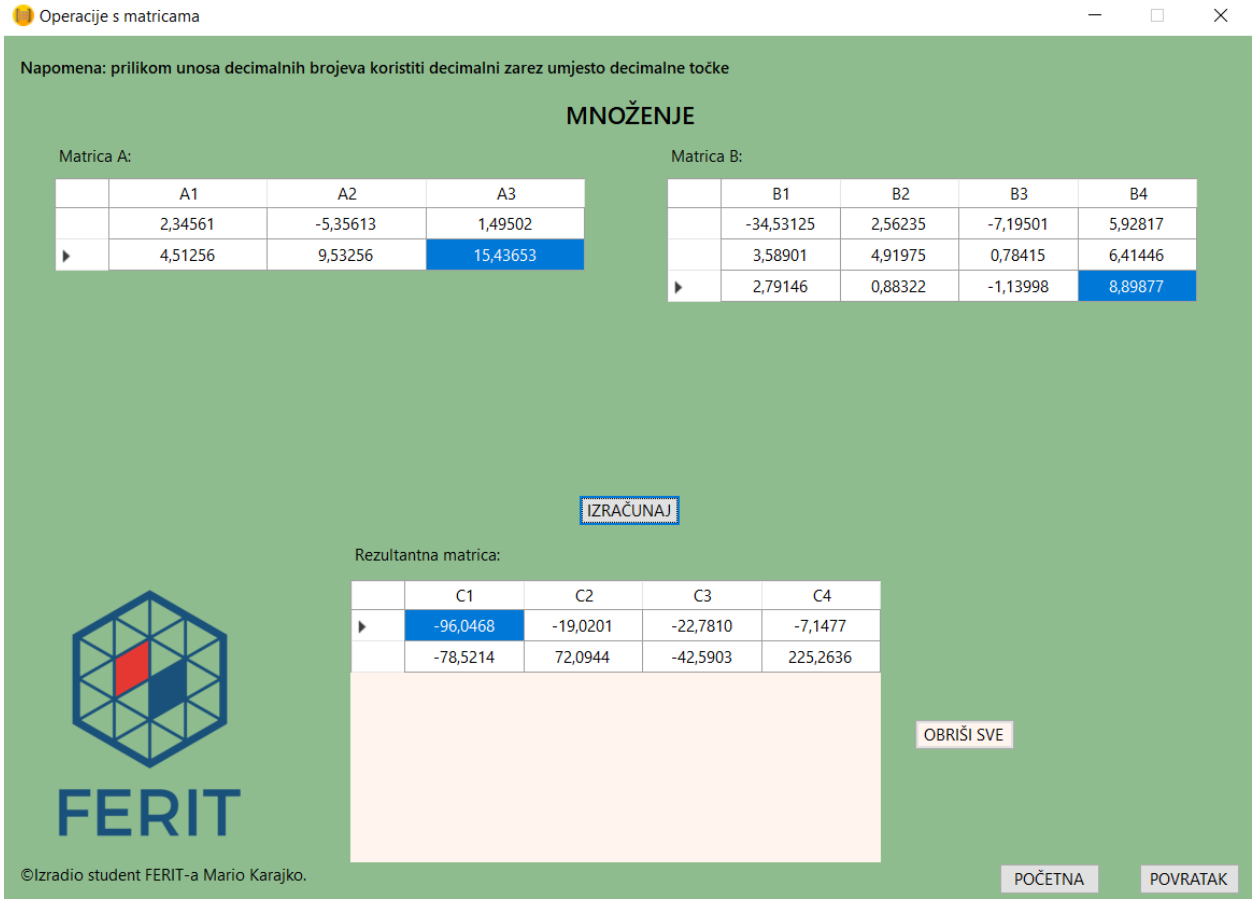
Slika 5.2. Odabir dimenzija matrica A i B

Klikom na gumb „POTVRDI“ pojavi se prozor prikazan na slici 5.3.



Slika 5.3. Prozor nul-matrica A i B

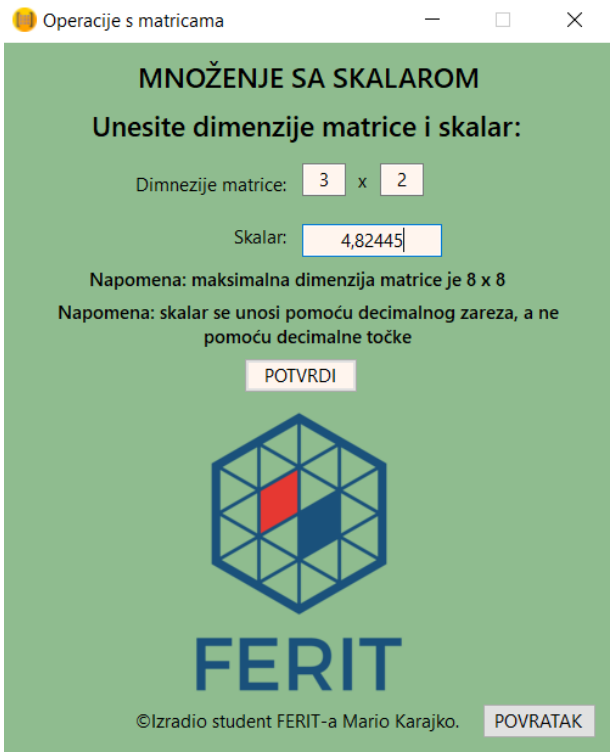
Nakon popunjavanja matrica A i B te klika na gumb „IZRAČUNAJ“ izračunava se rezultantna matrica C te na slici 5.4. mogu se vidjeti njezini elementi zaokruženi na četiri decimale.



Slika 5.4. Prikaz rezultantne matrice C nakon izvođenja operacije množenja

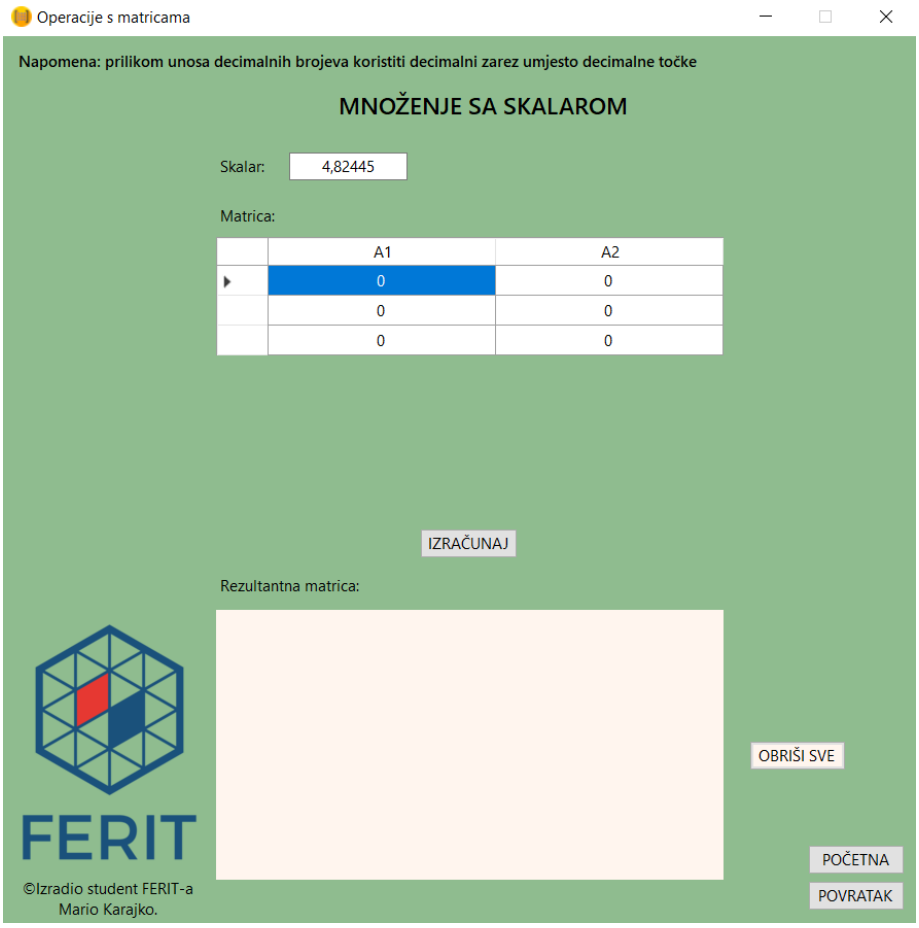
S ovim su prikazane tri od ukupno pet formi, a u nastavku će biti prikazane još i forme vezane za operaciju množenja matrice sa skalarom.

Upis dimenzija matrice i skalara može se vidjeti na slici 5.5.



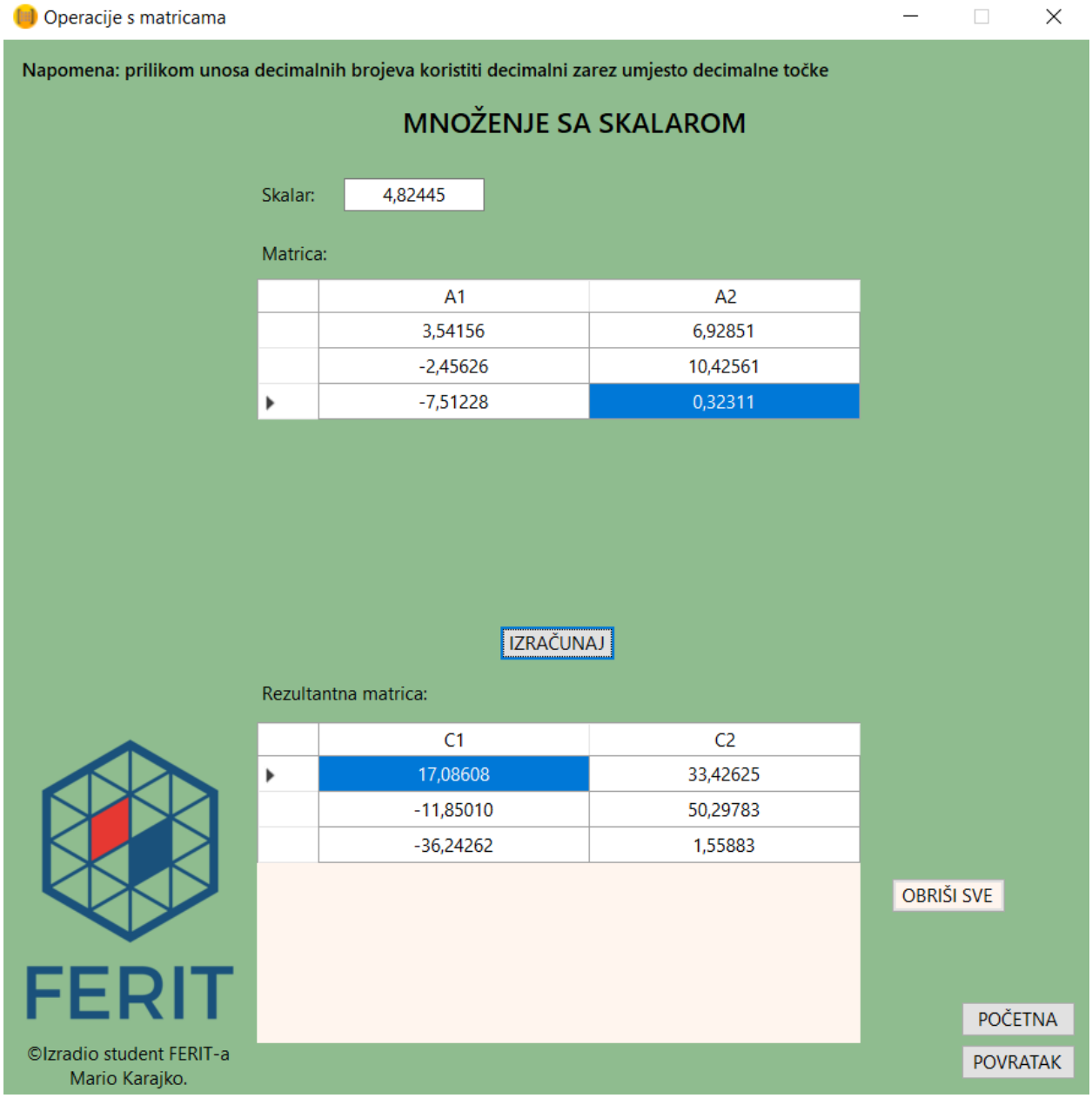
Slika 5.5. Upis dimenzija matrice i skalara

Klikom na gumb „POTVRDI“ pojavi se prozor prikazan na slici 5.6.



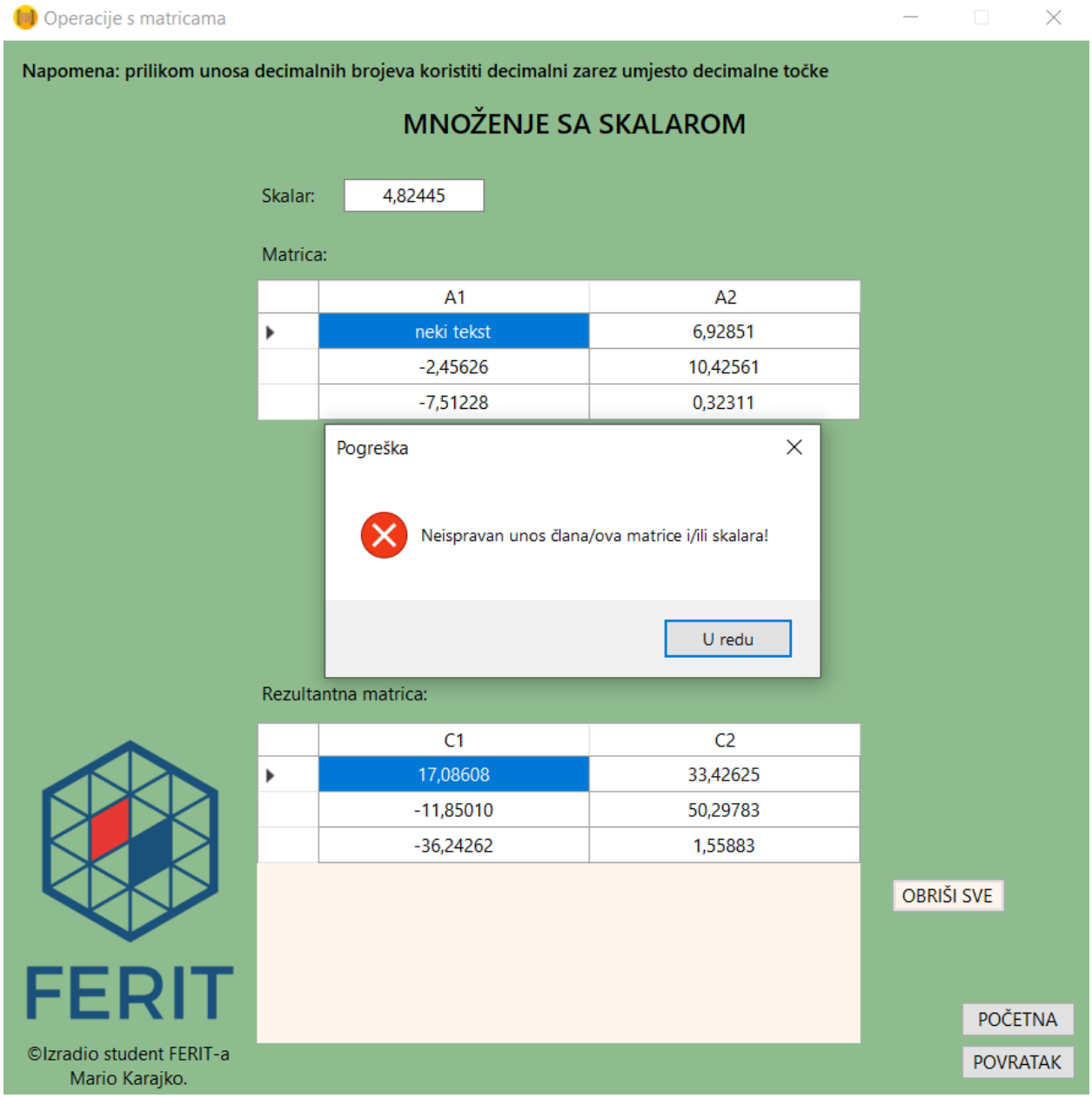
Slika 5.6. Prikaz prethodno unesenog skalara i početne nul-matrice

Nakon popunjavanja elemenata matrice i klika na gumb „IZRAČUNAJ“ izračunava se rezultantna matrica koja se može vidjeti na slici 5.7.



Slika 5.7. Prikaz rezultantne matrice nakon izvođenja operacije množenja matrice sa skalarom

Unosom neispravnog elementa matrice na poziciji [1, 1] te klikom na gumb „IZRAČUNAJ“ korisniku će iskočiti prozorčić s pogreškom kao što je vidljivo na slici 5.8.



Slika 5.8. Neispravan unos elementa matrice i prozorčić s pogreškom

Ovime je prikazan rad aplikacije. Operacije zbrajanja i oduzimanja izvode se na analogan način kao i operacija množenja pomoću unosa dimenzija matrica A i B te kasnijeg unosa elemenata tih matrica.

6. ZAKLJUČAK

Ciljevi postavljeni u zadatku završnog rada ostvareni su prvenstveno dobrom teorijskom pripremom prije početka same realizacije zadatka. Potrebno je bilo upoznati se s dotad nepoznatim Windows formama i svim njenim mogućnostima.

Na početku rada dana je teorijska, matematička podloga o samim matricama i operacija s njima. Nadalje, dan je kratak uvod o aplikaciji *Microsoft Visual Studio*, programskom jeziku C# te naposljetku o Windows formama nakon čega su opisani detaljni koraci prilikom kreiranja projekta unutar *Visual Studio* aplikacije. Pregled područja teme pokriven je s primjerima „*online*“ kalkulatora za računanje operacija s matricama te s primjerom mobilne aplikacije koja nudi i detaljniji prikaz koraka izvođenja operacije. Najzahtjevniji dio završnog rada čini sama izrada Windows desktop aplikacije koja se sastoji od ukupno pet Windows formi pomoću kojih je napravljen grafički prikaz operacija s matricama u programskom jeziku C#. Forme su objašnjene slijedno i to na način da je prvo prikazan izgled svake forme uz popis svih kontrola koje su prisutne unutar forme. Za svaku su formu objašnjeni svi programski kodovi, odnosno metode koje čine samu formu i na taj je način objašnjeno na koji način funkcionira svaka forma. Izgled konačne aplikacije prikazan je kroz nekoliko slika na kojima se može vidjeti princip korištenja same aplikacije te način na koji bi se aplikacija trebala koristiti, ali i moguće izvanredne situacije kao što su neispravni unosi elemenata matrice i slično.

Aplikacija je napravljena tako da matrice podržavaju cijele i decimalne brojeve što je ograničenje ukoliko korisnik želi izvršavati operacije s kompleksnim brojevima. Kako god, aplikacija može biti od velike koristi ukoliko je korisniku potrebna provjera je li dobro izračunao rezultantnu matricu za operacije zbrajanja, oduzimanja, množenja ili množenja sa skalarom.

LITERATURA

[1] D. Bakić, Linearna algebra, Školska knjiga, Zagreb, 2008.

[2] N. Elezović, Linearna algebra, Element, Zagreb, 2006.

[3] Microsoft, What is Visual Studio? [online], dostupno na: https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022 [pristupljeno 12. lipnja 2023.]

[4] Microsoft, A tour of the C# language [online], dostupno na: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ [pristupljeno 12. lipnja 2023.]

[5] Microsoft, What is .NET? [online], dostupno na: https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet [pristupljeno 12. lipnja 2023.]

[6] AltexSoft, The Good and the Bad of C# Programming [online], dostupno na: https://www.altexsoft.com/blog/c-sharp-pros-and-cons/ [pristupljeno 12. lipnja 2023.]

[7] Microsoft, Desktop Guide (Windows Forms .NET) [online], dostupno na: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-7.0 [pristupljeno 12. lipnja 2023.]

[8] Microsoft, Tutorial: Create a Windows Forms app with .NET [online], dostupno na: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/get-started/create-app-visual-studio?view=netdesktop-7.0 [pristupljeno 12. lipnja 2023.]

SAŽETAK

Glavni zadatak ovog završnog rada je napraviti Windows desktop aplikaciju koja će predstavljati grafički prikaz operacija s matricama. Pri pisanju programskog koda korišten je programski jezik C#, a kao razvojno okruženje korištena je *Microsoft Visual Studio* aplikacija. Također, korištene su Windows forme kao dio Microsoft .NET platforme za grafički prikaz ukupno pet formi. Prva forma predstavlja početni prozor aplikacije u kojem se pomoću padajućeg izbornika bira neka od sljedećih operacija: zbrajanje, oduzimanje, množenje te množenje sa skalarom. Druga forma predstavlja prozor u kojem korisnik upisuje dimenzije matrica A i B potrebne za operacije zbrajanja, oduzimanja i množenja. Treća forma predstavlja prozor za upis elemenata matrica A i B te joj je glavna zadaća izračunavanje rezultantne matrice klikom na gumb. Također, posjeduje i mogućnosti koje se izvode klikom na odgovarajući gumb, a podrazumijevaju: povratak na prethodnu formu, povratak na početnu formu i „čišćenje“ svih matrica na početno stanje. Četvrta se forma odnosi na prozor u kojemu korisnik upisuje dimenzije matrice i skalar koji su potrebni za operaciju množenja matrice sa skalarom. Zadnja, peta forma predstavlja unos elemenata matrice, a glavna zadaća joj je izračunavanje rezultantne matrice. Ona posjeduje sve mogućnosti kao i treća forma uz dodatak da korisnik u svakom trenutku može promijeniti vrijednost skalara koju je prethodno upisao u četvrtoj formi. Aplikacija je otporna na neispravne unose od strane korisnika, primjerice neispravan unos dimenzija matrice, neispravan unos skalara, neispravan unos elemenata matrice i tako dalje.

Ključne riječi: matrice, Microsoft Visual Studio, operacije s matricama, programski jezik C#, Windows forme

SUMMARY

GRAPHICAL DISPLAY OF OPERATIONS WITH MATRICES IN C# PROGRAMMING LANGUAGE

The main task of this final paper is to create a Windows desktop application that will represent a graphical display of operations with matrices. The programming language C# was used when writing the program code, and the *Microsoft Visual Studio* application was used as the development environment. Also, Windows Forms were used as part of the Microsoft .NET platform for graphical display of a total of five forms. The first form represents the initial window of the application in which one of the following operations is selected using the drop-down menu: addition, subtraction, multiplication and multiplication with a scalar. The second form represents a window in which the user enters the dimensions of matrices A and B needed for addition, subtraction and multiplication operations. The third form represents a window for entering the elements of matrices A and B, and its main task is to calculate the resultant matrix by clicking on the button. It also has options that can be performed by clicking on the corresponding button, which include: return to the previous form, return to the initial form and "cleaning" all matrices to their initial state. The fourth form refers to the window in which the user enters the dimensions of the matrix and the scalar required for the operation of multiplying the matrix with a scalar. The last, fifth form represents the entry of matrix elements, and its main task is to calculate the resultant matrix. It has all the same possibilities as the third form with the addition that the user can change the value of the scalar previously entered in the fourth form at any time. The application is resistant to incorrect inputs by the user, for example incorrect input of matrix dimensions, incorrect input of scalar, incorrect input of matrix elements and so on.

Keywords: matrices, Microsoft Visual Studio, operations with matrices, programming language C#, Windows Forms