Rješavanje determinante u programskom jeziku C++

Jukić, Grgur

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij računarstva

RJEŠAVANJE DETERMINANTE U PROGRAMSKOM JEZIKU C++

Završni rad

Grgur Jukić

Osijek, 2020.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

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

Osijek, 17.09.2020.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Grgur Jukić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3647, 16.09.2019.
OIB studenta:	10983666859
Mentor:	Doc.dr.sc. Tomislav Rudec
Sumentor:	Izv. prof. dr. sc. Alfonzo Baumgartner
Sumentor iz tvrtke:	
Naslov završnog rada:	Rješavanje determinante u programskom jeziku C++
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 2 razina
Datum prijedloga ocjene mentora:	17.09.2020.
Datum potvrde ocjene Odbora:	

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 23.09.2020.

Ime i prezime studenta:	Grgur Jukić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3647, 16.09.2019.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Rješavanje determinante 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

Tema završnog rada je izrada C++ aplikacije koja računa determinantu matrice do dimenzija 10x10 uz implementaciju grafičkog sučelja za korisnike. Standardni C ++ uključuje unos i ispis putem konzole, te ne uključuje nikakve pogodnosti za grafičko korisničko sučelje.

Cilj je proučiti Qt platformu i pripadajuće biblioteke, te uz pomoć dostupnih alata napraviti program pisan u C++ jeziku, koji ima prigodno grafičko sučelje. Omogućiti izvođenje na različitim softverskim i hardverskim platformama kao što su Linux, Windows, macOS i Android.

Aplikacija računa determinantu rekurzivno, omogućavajući korisniku da unese željenu veličinu matrice, kao i njene elemente, uz ograničenje da matrica nema veće dimenzije od 10x10, te jednim klikom izračuna determinantu matrice. Pri izradi je korišteno znanje stečeno na fakultetu iz kolegija „Objektno orijentirano programiranje“ i iz kolegija „Linearna algebra“. Pri izradi aplikacije korištena je knjiga Damira Bakića „Linearna Algebra“. [1]

Pomoću nje razrađena je logika i izabran algoritam koji će se koristiti za računanje determinante. Korišteno je i dodatno znanje koje je stečeno pri upoznavanju sa programom Qt. Koristeći prethodno navedeni software, omogućio sam da aplikacija ima interaktivno sučelje, te može biti korištena na različitim platformama, omogućavajući različitim korisnicima dosljednu kvalitetu usluge.

Prvi dio završnog rada bavit će se programskim jezikom, linearnom algebrom i tehnologijama koje su korištene pri izradi samog programa. Zatim će biti objašnjen kod koji računa determinantu matrice, nakon toga objašnjava se dizajn samog grafičkog sučelja za korisnike, te prikaz izračuna determinante za 4x4 i 7x7 kvadratnu matricu.

1.1. Zadatak završnog rada

Izraditi program pisan u programskom jeziku C++ koji u sebi sadrži stranu biblioteku za izradu grafičkog sučelja pomoću kojega korisnik može izabrati veličinu matrice, njene elemente, te pomoću navedenih unosa, izračunati determinantu. Program je izrađen u Qt-u, više platformskom *frameworku,* koji koristi C++, kompatibilnom s mnoštvom operativnih sustava i hardvera.

2. ALATI

U drugom poglavlju završnog rada govorit ćemo o alatima koji su korišteni za izradu aplikacije i kako im izgleda sintaksa. Prezentiran će biti i način upotrebe alata, ali ne previše detaljno. Za rješavanje determinante potrebno je i znanje iz linearne algebre, posebno iz teorije matrica.

2.1. Znanstveni dosezi - Algoritmi za računanje determinante matrice

Najčešći algoritmi koji se implementiraju zasigurno uključuju Leibnizove formule ili Laplaceove formule. Ovakvi pristupi su neučinkoviti za velike matrice. Broj potrebnih operacija za izračun determinante raste vrlo brzo sa povećanjem dimenzija same matrice. Na primjer, Leibnizova formula traži izračun n! produkta. Zbog napomenute neučinkovitosti morali su se razviti novi algoritmi i tehnike koje su učinkovitije za izračun determinante matrice.

Neke metode izračunavaju determinantu matrice zapisujući matricu kao umnožak matrica čije se determinante mogu lakše odrediti. Takve se tehnike nazivaju metodama razgradnje (engl. *decomposition methods*). Primjeri uključuju QR dekompoziciju, LU dekompoziciju i Choleskyevu dekompoziciju za pozitivno definitne matrice. Ove metode su reda O (n^3) što je veliko poboljšanje naspram Laplaceove koja je O (n!).

LU dekompozicija izražava matricu pomoću donje trokutaste matrice L, gornje trokutaste matrice U i permutacijske matrice P. Ova metoda faktorizacije matrice kao produkta dviju trokutastih matrica ima različite primjene, poput rješavanja sustava jednadžbi, koje je samo po sebi sastavni dio mnogih aplikacija poput pronalaženja struje u krugu i rješavanja diskretnih problema dinamičkog sustava; pronalaženje inverzne matrice i pronalaženje determinanta matrice. [2]

QR dekompozicija, predstavlja dekompoziciju matrice A u produkt pravokutne matrice Q i gornje trokutaste matrice R. [3]

2.2. C++

C++ je opće namjenski, objektno orijentirani programski jezik koji je stvorio Bjarne Stroustrup kao ekstenziju na C jezik. C++ obuhvaća i visoke i niske značajke programskih jezika, te se zbog toga smatra jezikom srednje razine. U početku zvan „C s klasama“ zbog sadržavanja svih svojstava C jezika, te nadodavanja koncepta klasa.

Jedno od glavnih svojstava C++ jezika je zbirka unaprijed definiranih klasa, koji su tipovi podataka koje je moguće instancirati više puta i omogućuje deklaraciju korisničko definiranih klasa. Same klase se dalje mogu prilagođavati funkciji svojih članova radi implementacije određenih funkcionalnosti. C++ omogućava preopterećenje različitih operatora kao što su operatori za aritmetiku, usporedbu, logički operatori ili operatori za manipuliranje bitovima. Uvode se bitni koncepti polimorfizma i nasljeđivanja, kao i virtualne i prijateljske funkcije, predlošci i *namespace-ovi.* [4]

Nasljeđivanje je jedan od najvažnijih koncepta objektno orijentiranog programiranja. Nasljeđivanje nam omogućava definiranje klase u smislu druge klase, što olakšava stvaranje i održavanje aplikacije. Također pruža priliku za ponovnu upotrebu funkcionalnosti koda i brzu implementaciju. [5]

Kada stvaramo klasu, umjesto pisanja potpuno novih članova podataka i funkcija članova, programer može odrediti da nova klasa treba naslijediti članove postojeće klase. Postojeća se klasa naziva osnovna klasa, a nova klasa, koju smo izveli, naziva se izvedenom klasom.

Ideja nasljeđivanja provodi „je“ odnos. Na primjer, sisavac je životinja, pas je sisavac, dakle pas je životinja, i tako dalje.

Polimorfizam je značajka OOP-a koja omogućuje da se objekt različito ponaša u različitim uvjetima. U C ++ imamo dvije vrste polimorfizma:

1) „*Compile time*“ polimorfizam- poznato kao statičko vezanje. Preopterećenje funkcija ili operatora primjer je *compile* time polimorfizma

2) „*Runtime*“ polimorfizam- poznato kao dinamičko vezivanje. Primjer je „*over-rideanje*“ funkcija. Kada podređena klasa deklarira metodu koja je već prisutna u roditeljskoj klasi, to se naziva „*over-rideanje* funkcije“, ovdje podređena klasa nadglašava roditeljsku klasu. [6]

Enkapsulirati znači osigurati da su "osjetljivi" podaci skriveni od korisnika. Kako bi smo to ostvarili, trebamo proglasiti varijable i atribute privatnima (nije im moguće pristupiti izvan klase). [7]

Ako želimo da drugi čitaju ili mijenjaju vrijednost privatnog člana, možemo pružiti javne načine dobivanja i postavljanja pomoću *aksesora* i *mutatora*. Dobra je praksa da atribute klase proglašavamo privatnim. Enkapsulacija povećava sigurnost koda i osigurava bolju kontrolu nad podacima, možemo promijeniti jedan dio koda bez utjecaja na druge dijelove.

Apstrakcija podataka znači prikazivanje samo bitnih podataka i skrivanje detalja. Apstrakciju možemo implementirati u C ++ koristeći klase. [8]

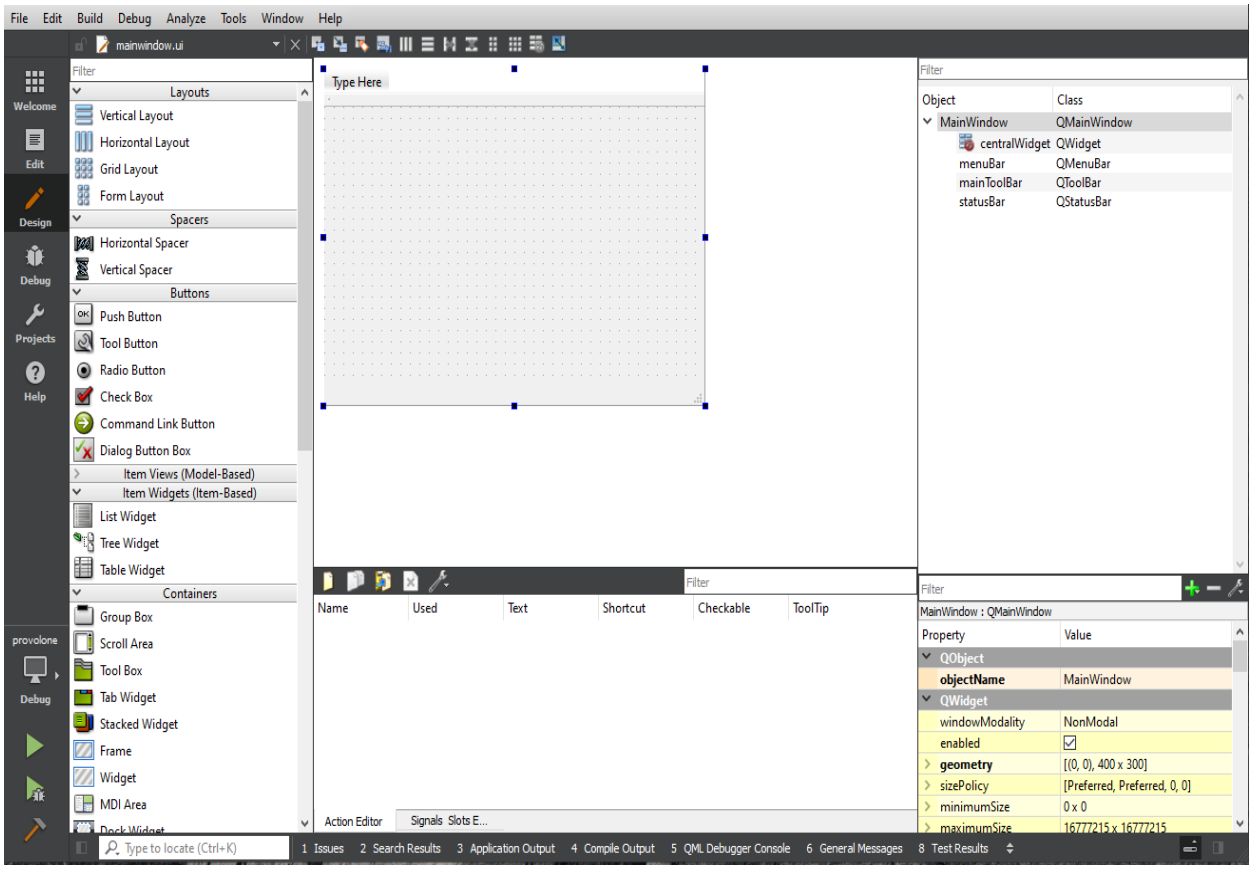
Klasa može odlučiti koji će član podataka biti vidljiv izvan svijeta, a koji nije.

2.3. Qt

Qt je *framework* za razvoj aplikacija na više platformi. Omogućava razvoj za različite tipove uređaja, na primjer, radne površine stolnih računala, ugrađene ili čak mobilne uređaje. Podržane platforme uključuju Linux, OS X, Windows, VxWorks, QNX, Android, iOS, BlackBerry, Sailfish OS i druge. [9]

Razvoj Qt-a započeli su 1990.godine norveški programeri Eirik Chambe-Eng i Haavard Nord. Njihova tvrtka, Trolltech, koja je prodavala Qt licence i pružala podršku, prošla je kroz nekoliko akvizicija tijekom godina kako bi postala tvrtka koja je danas. Danas se bivši Trolltech zove The Qt Company i u cjelokupnom vlasništvu Digia Plc., sa sjedištem u Finskoj. Iako je Qt Company glavni pokretač iza Qt-a, Qt je sada razvijen od strane većeg saveza: Qt Project. Sastoji se od mnogih tvrtki i pojedinaca širom svijeta i slijedi model meritokratskog upravljanja. [10]

Qt sam po sebi nije programski jezik. To je *framework* napisan na C ++. Pred-procesor, MOC (*Meta-Object Compiler*), koristi se za proširenje C ++ jezika sa značajkama poput *signala* i *slotova*. Prije same kompilacije, MOC analizira izvorne datoteke napisane u Qt-proširenom C ++ i iz njih generira standardne C ++ izvore koji su kompatibilni. Stoga sam *framework* i aplikacije / knjižnice koje ga koriste mogu se kompajlirati koristeći bilo koji standardni kompatibilni C ++ kompajler poput GCC-a, ICC-a, MinGW-a i MSVC-a.



Slika 2.1. Izgled prozora za razvoj grafičkog sučelja

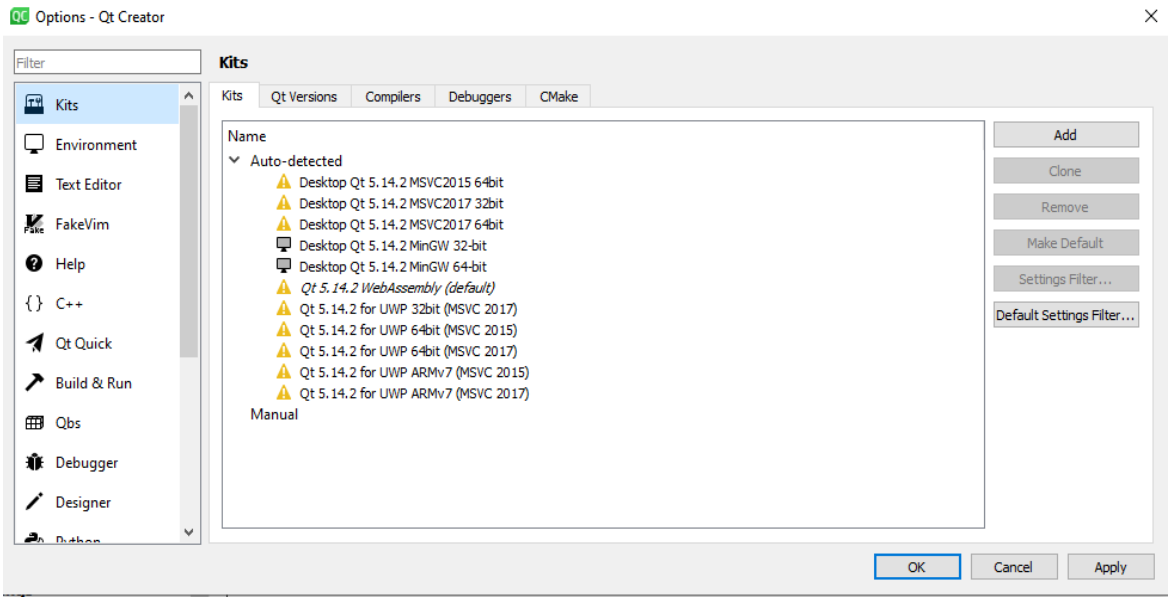
Qt dolazi sa vlastitim integriranim razvojnim okruženjem (IDE-om), pod nazivom Qt Creator. Radi na Linuxu, OS X-u i Windows-u i nudi inteligentno dovršavanje koda, isticanje sintakse, integrirani sustav pomoći i *debugger*. Uz Qt Creator, programeri koji koriste operacijski sustav Windows mogu koristiti i Qt's Visual Studio dodatak.

Pomoću Qt-a, grafička sučelja se mogu pisati izravno na C ++ jeziku pomoću modula *Widgets*. Qt također dolazi s interaktivnim grafičkim alatom pod nazivom Qt Designer koji funkcionira kao generator koda za GUI-ove temeljene na *widgetima*. Qt Designer se može koristiti samostalno, ali je također integriran u Qt Creatoru.

.

2.4. Qt SDK

SDK (software development kit) označava komplet alata za razvoj softvera. To je skup softverskih alata i programa koje programeri koriste za stvaranje aplikacija za određene platforme. SDK alati uključuju biblioteke, dokumentaciju, uzorke koda, procese i vodiče kojima se programeri mogu koristiti i integrirati u vlastite aplikacije. SDK-ovi su dizajnirani da se koriste za određene platforme ili programske jezike.



Slika 2.2. Prikaz paketa za razvoj softwarea

Qt SDK koristi prednosti Qt frameworka i njegovih dostupnih alata, kombinirajući ih sa alatima dizajniranim specifično za pojednostavljenje stvaranja aplikacija za desktop platforme poput Windowsa, Mac OS X i Linuxa kao i za stvaranje aplikacija za mobilne platforme. QT SDK omogućava jednostavnu instalaciju, za svaki od operacijskih sustava ima jedan instalacijski paket koji u sebi sadrži sve potrebno za programiranje i pokretanje aplikacija. Sadrži alate kao što su Qt Creator, Qt Designer i Qt Quick Designer, kao i API-je za mobilne aplikacije.

2.5. Linearna algebra- svojstva matrice

Linearna algebra je grana matematike koja proučava vektore, vektorske prostore, linearne operatore i sustave linearnih jednadžbi. Matrice omogućuju jednostavan zapis i rješavanje sustava linearnih jednadžbi. U linearnoj algebri determinanta je skalar, korisna vrijednost koja se može izračunati iz elemenata kvadratne matrice.

Determinanta matrice A, označava se det (**A**), det**A** ili | **A** |. U slučaju kvadratne matrice koja ima dimenzije 2x2, determinantu računamo pomoću formule 1.1.

$$|A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$$

Formula 1.1. Izračun determinante kvadratne matrice veličine 2x2

Slično tome, ako pretpostavimo da imamo 3x3 kvadratnu matricu A, želimo koristiti formulu za njenu determinantu.

$$\begin{aligned} |A| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} &= a\begin{vmatrix} e & f \\ h & i \end{vmatrix} - b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix} \\ &= aei + bfg + cdh - ceg - bdi - afh. \end{aligned}$$

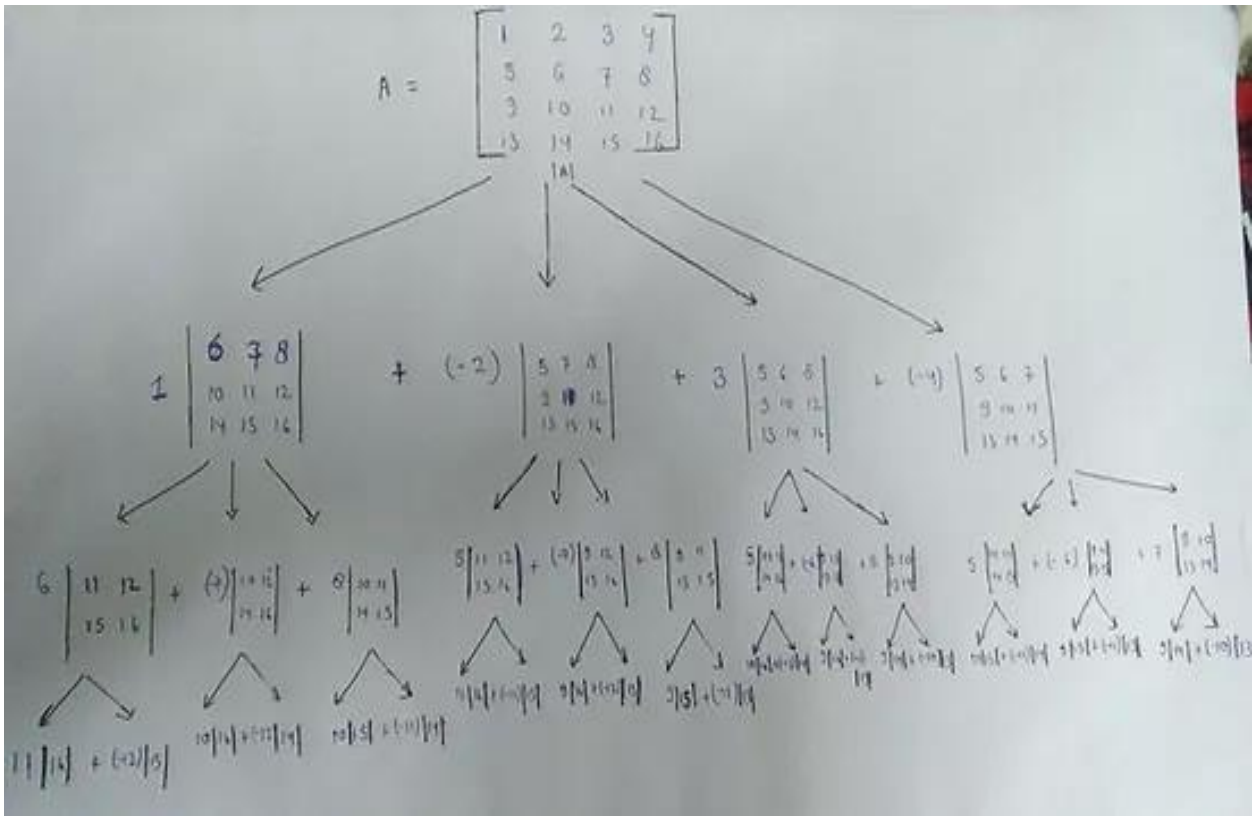
Formula 1.2. Izračun determinante kvadratne matrice veličine 3x3

Isti se postupak koristi za pronalazak determinanti matrica većeg stupnja (4x4, 5x5 i tako dalje). Aplikacija za izračun determinante podržava sve kvadratne matrice koje imaju dimenzije manje ili jednake 10x10.

Kako bi izračunali determinante kvadratnih matrica većih dimenzija, moramo koristiti rekurziju.

Ukoliko korisnik unese matricu većih dimenzija, rekurzivno ćemo „rastavljati“ matricu. Izračuna se determinanta podmatrice, te se množi sa vanjskom vrijednošću.

Uzimajući u obzir 4x4 matricu, možemo promatrati rekurzivni obrazac koji dijeli matricu na podmatrice i iz tih podmatrica računa determinantu. Postupak predočen na slici 2.3. zove se Laplaceov razvoj.

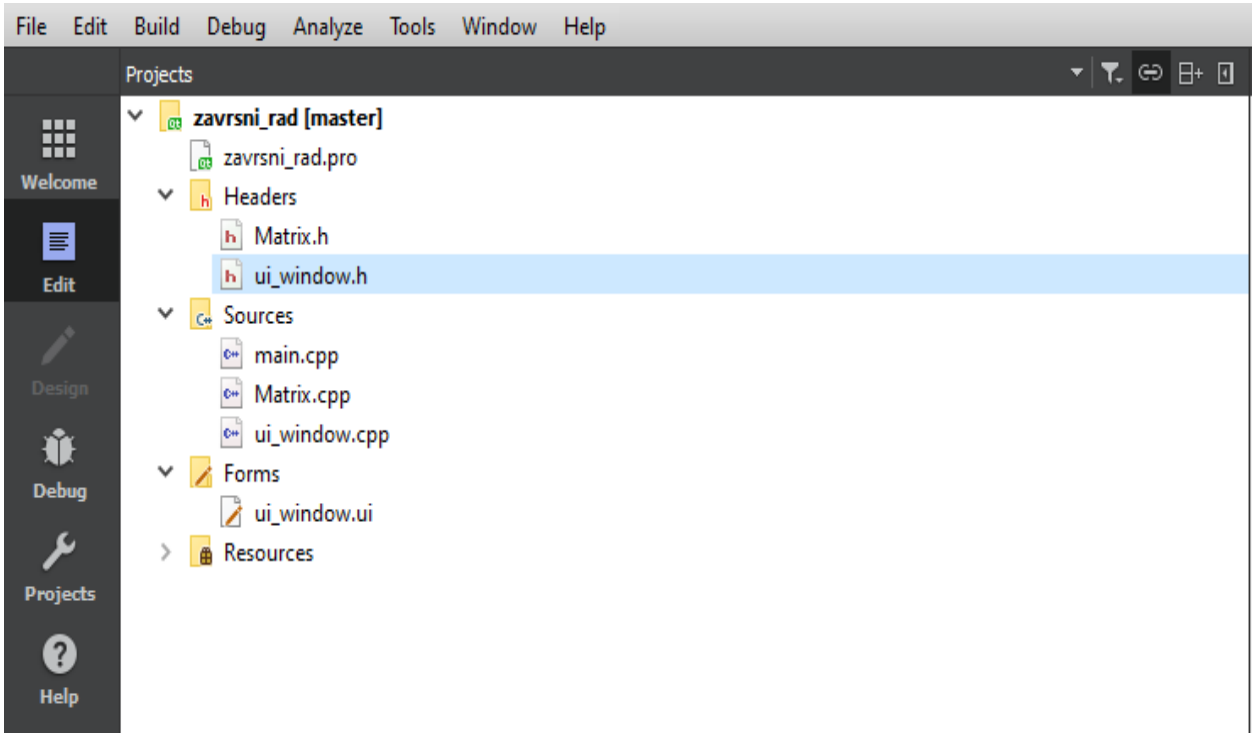


Slika 2.3. Primjer razlaganja matrice 4x4 pomoću Laplaceovog razvoja

3. STRUKTURA QT APLIKACIJE

3.1. Struktura programa

Konačan projekt za izračun determinante matrice sa prikladnim grafičkim sučeljem sastoji se od pro datoteke, datoteke zaglavlja za matricu, izvorne datoteke i prozora sa kojim ćemo raditi. Na kraju imamo ui datoteku u kojoj smo dizajnirali grafičko sučelje i spajali funkcionalnosti.



Slika 3.1. Prikaz sastavnica projekta

3.2. Pro datoteka (project datoteka)

Svrha datoteka sa pro ekstenzijom je da izlista popis izvornih datoteka koje su uključene u projekt. Budući da se Qmake koristi za izgradnju Qt-a i pridruženih alata, Qt vrlo dobro poznaje i može stvoriti pravila za pozivanje moc, uic i rcc-a. Kao rezultat toga, sintaksa je vrlo sažeta i lako se uči.

Meta-Object Compiler (moc) je program koji obrađuje Qt-ove C ++ ekstenzije. Moc čita datoteku zaglavlja C ++. Ako pronađe jednu ili više deklaracija klase koje sadrže makro Q_OBJECT, stvara izvornu datoteku C ++ koja sadrži meta-objektni kod za te klase koji je potreban za mehanizam signala i slotova i informacije o vrsti izvođenja. Izvorna datoteka C ++ generirana moc-om mora se sastaviti i povezati s implementacijom klase.

Kompilator korisničkog sučelja. (uic) čita datoteku s definicijom korisničkog sučelja formata XML (.ui) koju generira Qt Designer i stvara odgovarajuću C ++ datoteku zaglavlja.

Resursni kompilator (rcc) je alat koji se koristi za umetanje resursa u Qt aplikaciju tijekom procesa sastavljanja. Djeluje generiranjem izvorne datoteke C ++ koja sadrži podatke navedene u datoteci Qt resursa (.qrc).

```
#-------------------------------------------------
#
# Project created by QtCreator 2020-06-03T14:28:58
#
#-------------------------------------------------

QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = zavrsni_rad.pro
TEMPLATE = app

# The following define makes your compiler emit warnings if you use
# any feature of Qt which has been marked as deprecated (the exact warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if you use deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0

CONFIG += c++11

SOURCES += \
        Matrix.cpp \
        main.cpp \
        ui_window.cpp

HEADERS += \
        Matrix.h \
        ui_window.h

FORMS += \
    ui_window.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target
```

Slika 3.2. Prikaz zavrsni_rad.pro u Qt Creatoru

Za projekte aplikacija ili lib-a najčešće se koriste sljedeće varijable:

- HEADERS određuje datoteke C ++ zaglavlja (.h) filea.
- SOURCES određuje datoteke za implementaciju C ++ (.cpp) projekta.
- FORMS određuje Qt Designer ui datoteke koje treba obraditi uic.
- DEFINES specificira C ++ simbole pretprocesora koji bi trebali biti unaprijed definirani.
- CONFIG određuje različite konfiguracije projekta i mogućnosti prevoditelja.
- QT određuje Qt module koji se koriste u projektu. (Zadana je jezgra gui, što odgovara QtCore i QtGui modulima.)
- VERSION određuje broj verzije ciljne biblioteke.

TARGET određuje osnovno ime ciljne izvršne datoteke ili biblioteke, isključujući bilo koji nastavak, prefiks ili broj verzije.

3.3. Ui_window datoteka zaglavlja

```
#include <QMainWindow>
#include <QPushButton>
#include <QStackedLayout>
#include <QLineEdit>
#include <QLabel>
#include <QFileDialog>


namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow {
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = nullptr);
    MainWindow(QString title, int n);
    ~MainWindow();

public slots:
   void calculateDeterminant(QLabel *label);
   void resizeMatrix(QLineEdit* dimensions, QWidget *matrixFrame);
   void toDarkTheme(QWidget *Widget,QWidget *Frame,QWidget *determinant);
   void toFeritTheme(QWidget *Widget,QWidget *Frame,QWidget *determinant);
private:
    Ui::MainWindow *gui;
    QWidget *central;
    QVector<QVector<QLineEdit*>> array;
    QGridLayout *grid;
    int n;
    QLabel label;


};
```

Slika 3.3. Prikaz ui_window.h u Qt Creatoru

Datoteka zaglavlja je datoteka s ekstenzijom h koja sadrži deklaracije funkcija i definicije makronaredbi koje se trebaju podijeliti između nekoliko izvornih „*source*“ datoteka. Zatražujemo korištenje datoteka zaglavlja u programu, koristeći #include naredbu. Uključivanje datoteke zaglavlja jednaka je kopiranju sadržaja datoteke zaglavlja, ali to se ne radi jer nije dobro kopirati sadržaj datoteke zaglavlja u izvornim datotekama, posebno ako imaju više izvornih datoteka u programu. Jednostavna praksa u C ++ programima je da sve konstante, makronaredbe, globalne varijable i funkcijske prototipe držimo u datotekama zaglavlja i uključimo tu datoteku zaglavlja gdje god je potrebno.

Moramo uključiti QMainWindow klasu koja nam daje *framework* za građenje korisničkog sučelja. Služi za menadžment središnjeg, glavnog, prozora te automatski postavlja svoj zadani raspored na koji možemo dodavati različite funkcionalnosti.

QPushButton klasa nam omogućuje korištenje tipke za naredbe. Jedan od češće korištenih *widgeta* u bilo kakvom grafičkom sučelju. Kliknite tipku kako bih zapovjedili računalu da napravi određenu akciju.

QLineEdit nam omogućava da unesemo i uređujemo tekst, sadrži kolekciju funkcija za uređivanje, uključujući *undo* i *redo*, kopiranje i lijepljenje.

QLabel se koristi za prikaz teksta ili slike. Nije pružena funkcionalnost interakcije korisnika. Vizualni izgled naljepnice može se konfigurirati na različite načine.

Imamo auto generiran namespace zvan gui od strane Qt-a. Koristi se za grupiranje automatski generiranih prozora u jednom name-spaceu. Pomaže u razlikovanju između klase ui koju generira dizajnerska ui datoteka i klase koja implementira njenu funkcionalnost. *Namespace-ovi* nam omogućuju grupiranje imenovanih entiteta koji bi u protivnom imali globalni opseg u uže domete, dajući im manji domet prostora imena.

Imamo auto generiranu klasu MainWindow koja nasljeđuje javna svojstva klase QMainWindow koja je zadužena za kreiranje glavnog prozora sučelja. Q_OBJECT je makro koji deklarira svoje signale i slotove, mehanizme koje koristimo za komunikaciju između objekata. Signal se emitira kada se dogodi određeni događaj. Slot je funkcija koja se poziva kao odgovor na određeni signal.

U javnom dijelu klase imamo eksplicitni konstruktor kojem dajemo pokazivač na QWidget. =nullptr znači da ako ne predamo argument, pokazivač-roditelj će biti postavljen na NULL. Korištenjem ključne riječi *explicit* sprječavamo implicitno dodavanje vrijednosti. Koristimo kao zaštitu jer implicitno dodavanje vrijednosti može prouzročiti greške.

Pod slotovima imamo definirane metode calculateDeterminant**,** resizeMatrix, toDarkTheme, toFeritTheme koje se pozivaju kada imamo nekakvu akciju (klik na tipku). Slotovi su normalne C++ funkcije i mogu se pozivati kao takve, jedina razlika je u tome da signali moraju biti spojeni na njih, učitavaju int koji stoji u našem labelu kao argument.

Metoda za promjenu veličine kvadratne matrice nam omogućava postavljanje dimenzija kvadratne matrice na željenu dimenziju, primajući kao argument integer koji stoji na line editoru.

Metode toDarkTheme i toFeritTheme omogućavaju promjenu vizualnog izgleda sučelja. Jedna tema predstavlja boje FERIT-a(crvenu, bijelu i plavu). Druga je tamnija tema, koja je manje intenzivna za oči, tamnije teme se inače stavljaju u svakoj modernoj aplikaciji.

U privatnom djelu, deklariramo pokazivače na auto generiranu klasu MainWindow, i QWidget klasu za sve objekte korisničkog sučelja.

QVector je predložak klasa koja nam omogućuje dodavanje dinamičkog polja koji će nam služiti za matrice različitih dimenzija.

Na kraju imamo cjelobrojnu varijablu n koja će nam prikazivati veličinu unesene matrice.

3.4. Matrix datoteka zaglavlja

```
#ifndef MATRIX_H
#define MATRIX_H

#include <cstdlib>
#include <iostream>
#include <cmath>
#include <QDebug>

class Matrix {
private:
    int size;
    float **matrix;

    float** reduceSize(float** matrix, int size, float x, float y);
    float calculateRecursively(float** matrix, int size);

public:
    Matrix(int size);

    int getSize();

    void setElement(int row, int column, float value);
    float getElement(int row, int column);

    float calculateRecursively();


};

#endif
```

Slika 3.4. Prikaz Matrix.h u Qt Creatoru

- Zaglavlje <cstdlib> definira nekoliko funkcija opće namjene, uključujući dinamičko upravljanje memorijom, stvaranje slučajnih brojeva, komunikaciju s okolinom, cijelu aritmetiku, pretraživanje, sortiranje i pretvaranje.
- Zaglavlje <iostream> je datoteka koja sadrži sve funkcije programa poput cout, cin itd. i #include govori pretprocesoru da u program uključi te datoteke zaglavlja.
- Zaglavlje <cmath> deklarira skup funkcija za računanje uobičajenih matematičkih operacija i transformacija
- Zaglavlje <QDebug> pruža izlazni tok za ispravljanje pogrešaka. QDebug se koristi kad god programer mora napisati podatke o uklanjanju pogrešaka ili praćenja na uređaj, datoteku, string ili konzolu.

Klasa matrix se sastoji od :

- Privatne cjelobrojne varijable(int size) koju ćemo koristiti pri unosu veličine kvadratne matrice.
- Pokazivač matrix na dvodimenzionalno polje, omogućit će nam da poslije dinamički alociramo memoriju za elemente polja.
- Metoda reduceSize() koja nam omogućuje smanjivanje veličine matrice sve dok nemamo najmanji oblik koji možemo izračunati. Vraća novu, manju matricu, točnije pokazivač na dvodimenzionalno polje. Poziva se u metodi calculateRecursively(), dio rekurzije.
- Metoda calculateRecursively() prima matricu i veličinu matrice, i pokreče postupak računanja determinante. Vraća cjelobrojnu vrijednost koja predstavlja determinantu dane matrice.
- Pod javnim članovima imamo konstruktor koji kao argument prima veličinu kvadratne matrice.
- Metodu getSize() koja nam vraća veličinu matrice.
- Metoda setElement() prima dva broja koja predstavljaju redak i stupac elementa čiju vrijednost želimo postaviti. Prima decimalne brojeve.
- Metoda getElement() vraća element koji je korisniku zanimljiv, daje mu točan stupac i redak elementa koji ga zanima.
- Metoda calculateRecursively() omogućuje rekurziju, poziva ponovo metodu za računanje determinante sa danim redom i stupcem elementa polja, uključujući i veličinu matrice.

3.5. Main izvorna datoteka

```
#include "ui_window.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow window("Grgur's Determinant calculator",1);
    window.show();

    return a.exec();
```

Slika 3.5. Prikaz main.cpp u Qt Creatoru

U main.cpp datoteci kreiramo naše grafičko korisničko sučelje, u ovom slučaju jedinstveni prozor kojem dajemo ime „Grgur's Determinant Calculator“ i postavljamo da, prvo otvaranje prozora prikazuje 1x1 matricu i prikazuje jedini element koji je spreman za unos vrijednosti.

<QApplication> uključuje definiciju klase QApplication. U svakoj aplikaciji koja koristi Qt mora postojati točno jedan objekt QApplication. QApplication upravlja raznim resursima koji se primjenjuju na cijeloj aplikaciji, kao što su zadani font i skraćenice.

Glavna funkcija je ulazna točka u program. Kada koristimo Qt, main () treba izvršiti neku vrstu inicijalizacije prije nego što kontrolu predamo u Qt knjižnicu, koja potom programu govori o događajima korisnika putem događaja.

argc je broj argumenata naredbenog retka, a argv je polje argumenata naredbenog retka. Ovo je C / C ++ značajka. Nije specifičan za Qt; međutim, Qt mora te argumente obraditi (vidjeti sljedeće).

Widget nikad nije vidljiv poslije njegove kreacije, zato moramo pozvati window.show().

Return a.exec() je mjesto gdje main() prenosi kontrolu na Qt, a exec() će se vratiti kad aplikacija izađe. U exec(), Qt prima i obrađuje događaje korisnika i sustava i prosljeđuje ih odgovarajućim *widgetima*.

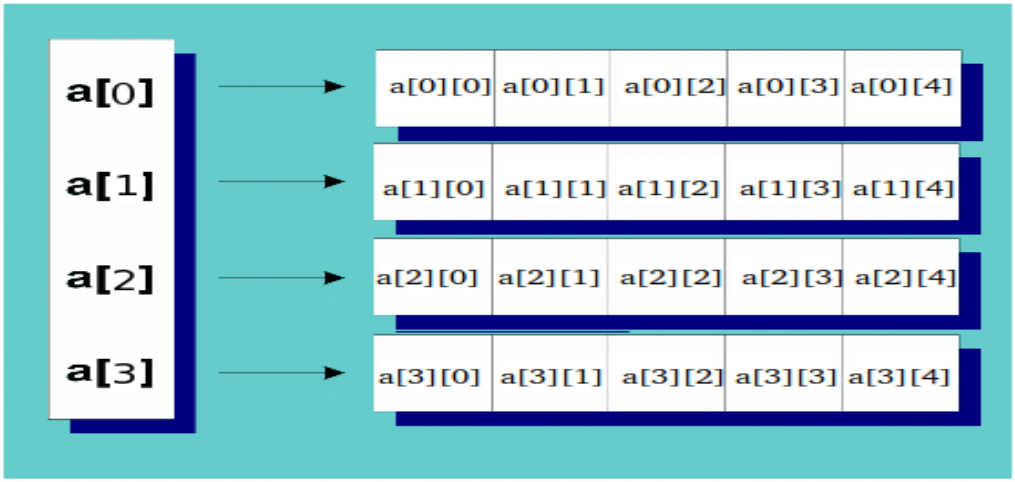
3.6. Matrix izvorna datoteka

U matrix.cpp datoteci definiramo sve metode potrebne za rad sa matricom i računanje determinante matrice.

```
#include "Matrix.h"

Matrix::Matrix(int size){

    this->size = size;

    matrix = new float*[size];
    for(int row=0;row<size;row++){
        matrix[row] = new float[size];
    }
}
```

Slika 3.6. Prikaz kreiranja dvodimenzionalnog polja koji predstavlja matricu proizvoljne veličine u Matrix.cpp izvornoj datoteci

Prvo imamo definiciju konstruktora Matrix koji prima vrijednost koja predstavlja željenu veličinu matrice. Dinamična dvodimenzionalna matrica je u osnovi niz pokazivača na polje. Možemo je inicijalizirati pomoću petlje for.



Slika 3.7. Prikaz inicijalizacije dvodimenzionalnih polja

Imamo definiciju naše metode getSize() koja služi za vraćanje veličine polja. Implementacija metode setElement() koja dodjeljuje željenom elementu unesenu vrijednost. Unosimo prvo redak pa stupac elementa kojem želimo promijeniti vrijednost, zatim vrijednost koju želimo.

Metoda getElement() vraća na unos retka i stupca, vrijednost koja se nalazi na tom mjestu.

```
int Matrix::getSize(){
    return size;
}


void Matrix::setElement(int row, int column, float value){
    this->matrix[row][column]=value;
}

float Matrix::getElement(int row, int column){
    return matrix[row][column];
}
```

Slika 3.8. *Akcesori i mutatori*

Metoda reduceSize() kreira novo dvodimenzionalno polje koje predstavlja novu smanjenu matricu od originalne. Inicijaliziramo ju, te prolazimo kroz matricu gledajući, ako elementi nisu elementi prvog retka i istog tog stupca, vrijednost na tom retku i stupcu bit će jednaka vrijednosti elementa na tom retku i stupcu originalne matrice te, nakon toga, povećavamo stupac za jedan. Sljedeća petlja govori, da ako nismo na kraju retka, povećavamo redak za jedan. Na kraju vraćamo dobivenu matricu.

```
float** Matrix::reduceSize(float** matrix, int size, float x, float y){
    float** newMatrix = new float*[size-1];
    for(int row=0;row<(size-1);row++){
        newMatrix[row] = new float[size-1];
    }
    int newColumn=0,newRow=0;
    for(int row=0;row<size;row++){
        for(int column=0;column<size;column++){
            if(row!=x && column!=y){
                newMatrix[newRow][newColumn]=matrix[row][column];
                newColumn++;
            }
        }
        if(row!=x){
            newRow++;
        }
        newColumn=0;
    }
    return newMatrix;
}
```

Slika 3.9. Prikaz metode za rastavljanje matrice na podmatrice

Metoda calculateRecursively() prvo provjerava je li unesena vrijednost za veličinu matrice jedan, ukoliko jest, vrati prvi element polja, pošto je njegova vrijednost ujedno i determinanta 1x1 matrice. Ukoliko je neki drug broj unesen za veličinu matrice, računamo rekurzivno. Uvodimo cjelobrojnu vrijednost sum koja će nam predstavljati sumu svih determinanti manjih matrica, te u konačnici determinantu naše prvotno dane matrice.

```
float Matrix::calculateRecursively(float** matrix, int size){
    if(size==1){
        return matrix[0][0];
    } else {
        float sum=0;
        for(int column=0;column<size;column++){
            sum+=pow(-1,column)*matrix[0][column]*calculateRecursively(reduceSize(matrix,size,0,column),size-1);
        }
        return sum;
    }
}
```

Slika 3.10. Rekurzivna metoda za izračun determinante

3.7. Ui_window izvorna datoteka

```
#include "ui_window.h"
#include "ui_mainwindow.h"
#include "Matrix.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    gui(new Ui::MainWindow)
    {
        gui->setupUi(this);
     ;
    }
```

Slika 3.11. Prvi dio mainwindow source datoteke u Qt Creatoru

Klasu koju generira kompajler korisničkog sučelja naziva se također MainWindow, ali pripada namespaceu Ui, pa je njezino puno ime Ui::MainWindow. To nije isto što i klasa MainWindow koja nasljeđuje QMainWindow. Međutim, klasa izvedena iz QmainWindow ima pokazivač člana nazvan ui na Ui :: MainWindow, koji je inicijaliziran u konstruktoru. Svi članovi Ui :: MainWindow su javni i stoga im je u potpunosti dostupan unutar MainWindowa.

QMainWindow (roditelj) poziva konstruktor za nadklasu MainWindow, a Ui kreira novi ui član

```
#include "ui_window.h"
#include "ui_mainwindow.h"
#include "Matrix.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    gui(new Ui::MainWindow)
    {
        gui->setupUi(this);

    }

MainWindow::MainWindow(QString title, int n){
    this->setWindowTitle(title);
    this->n = n;

    QWidget *Frame = new QWidget(this);
    QGridLayout *GridLayout = new QGridLayout();
    QWidget *Widget = new QWidget;
    Widget->setLayout(GridLayout);
    setCentralWidget(Widget);
    GridLayout->addWidget(Frame);

    QVBoxLayout *mainLayout = new QVBoxLayout;
    QStackedLayout *stackLayout = new QStackedLayout;
    QWidget *determinant = new QWidget();
    Frame->setLayout(mainLayout);
```

Slika 3.12. Inicijalizacijska lista za sučelje, kreiranje widgeta i postavljanje layouta za determinantu

Funkcija MainWindow prima ime naše aplikacije kao i cjelobrojnu vrijednost koja predstavlja veličinu kvadratne matrice koja će biti prikazana pri pokretanju aplikacije

Pomoću this pokazivača pristupamo točno ovim vrijednostima u memoriji.

Frame je podatak tipa QWidget koji nam služi za kreiranje okvira, odnosno glavnog prozora.

GridLayout služi za raspored Widgeta u prostoru, tipa je QGridLayout. Koji zauzima prostor koji mu je dostupan (prema nadređenom rasporedu ili nadređenom widgetu), dijeli ga na retke i stupce i svaki widget kojim upravlja postavlja u ispravnu ćeliju.

setCentralWidget postavlja Widget kao sredisnji element glavnog prozora.

mainLayout stoji kao glavni prozor, QVBoxLayout nam omogućava postavljanje widgeta vertikalno, odnosno omogućuje nam kreiranje vertikalnog rasporeda izgleda kutije.

stackLayout je tipa QStackedLayout koji nam omogućava slojevito postavljanje widgeta. Samo jedan widget je vidljiv u isto vrijeme.

Frame widgetu se postavlja layout u mainLayout.

```
QPalette pal = palette();
pal.setColor(QPalette::Background, Qt::white);
determinant->setPalette(pal);
determinant->show();
determinant->setAutoFillBackground(true);

QPalette pal2 = palette();
pal2.setColor(QPalette::Background, Qt::blue);
Frame->setPalette(pal2);
Frame->show();
Frame->setAutoFillBackground(true);

QPalette pal3 = palette();
pal3.setColor(QPalette::Background, Qt::red);
central->setPalette(pal3);
central->show();
central->setAutoFillBackground(true);

QFont f("Arial",15, QFont::Bold);
QFont d("Arial",13, QFont::Bold);
```

Slika 3.13. Dizajn grafičkog sučelja, pozadinske boje i fontovi

U ovom dijelu koda kreiramo tri varijable pal 1, pal 2 i pal 3 koja su tipa QPallete. QPallete nam omogućava promjenu boja različitih objekata u aplikaciji. Uzimamo bijelu, plavu i crvenu koje su boje našeg fakulteta, te ih postavljamo za pozadinske boje određenih okvira naše aplikacije.

Varijable f i d su tipa QFont, koji nam omogućava podešavanje teksta unutar aplikacije. Izabiremo font i veličinu fonta.

```
QWidget *matrixFrame = new QWidget(determinant);
QVBoxLayout *determinantLayout = new QVBoxLayout;
stackLayout->addWidget(determinant);
determinant->setLayout(determinantLayout);
determinantLayout->addWidget(matrixFrame);
mainLayout->addLayout(stackLayout);


QWidget *pushButton = new QWidget;
QLabel *label =new QLabel;
label->setText("App for solving the determinant of a matrix. Uses a recursion algorithm to solve the problem.");
label->setMinimumWidth(100);
label->setMinimumHeight(100);
QLabel *descriptionDimensions = new QLabel("Insert dimensions of the matrix (can't be bigger than 10x10) :");
QLabel *descriptionFaculty = new QLabel("Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek (FERIT Osijek)");
QLabel *descriptionName = new QLabel("Grgur Jukić");
QLineEdit *size = new QLineEdit();
QPushButton *exitButton = new QPushButton("Quit the application");
QPushButton *resizeButton = new QPushButton("Resize the matrix");
QPushButton *calculateButton = new QPushButton("Calculate the determinant");
QVBoxLayout *verticalButtonLayout = new QVBoxLayout;
QHBoxLayout *horizontalButtonLayout = new QHBoxLayout;
QWidget *pushButtonTop = new QWidget;

verticalButtonLayout->addWidget(label);
pushButton->setLayout(verticalButtonLayout);
determinantLayout->addWidget(descriptionFaculty);
descriptionFaculty->setFont(f);
descriptionName->setFont(d);
determinantLayout->addWidget(descriptionName);
verticalButtonLayout->setSizeConstraint(QLayout::SetFixedSize);
verticalButtonLayout->addWidget(descriptionDimensions);
verticalButtonLayout->addWidget(size);
verticalButtonLayout->addWidget(resizeButton);
verticalButtonLayout->addWidget(calculateButton);
pushButtonTop->setLayout(horizontalButtonLayout);
horizontalButtonLayout->addWidget(pushButton);
verticalButtonLayout->addWidget(exitButton);
```

Slika 3.14. Kreiranje različitih tipki i naljepnica, postavljanje rasporeda u sučelju

Kreiramo više tipki tipa QPushButton koje će nam služiti za postavljanje različitih zahtjeva aplikaciji. Imamo tipku za izračun determinante, promjenu veličine matrice i izlazak iz aplikacije. Svaka tipka ima određeni tekstualni zapis koji možemo vidjeti unutar navodnika *widgeta* QPushButton.

Kreiramo i par tekstualnih naljepnica koje služe za dodatne informacije koje želimo pružiti korisniku. Takvi naljepnice u svojem imenu počinju sa *description*. Korisniku se govori da u prostor za unos vrijednosti unese željenu veličinu matrice, upozoravajući ga da matrica ne smije biti veća od 10x10. Ostale naljepnice se odnose na ime fakulteta kao i autora aplikacije.

Poseban tip koji imamo je LineEdit koji nam omogućuje unos cjelobrojnog broja koji će se koristiti kao željena dimenzija matrice od strane korisnika.

Nakon toga, sve što je kreirano postavlja se u odgovarajući raspored. Određene informacije i mogućnosti poput tipki pripadaju odgovarajućem rasporedu. Sve tipke, uključujući i prostor za unos idu u poseban raspored.

```
size->setText("1");
grid = new QGridLayout();
for(int row=0;row<n;row++){
   array.append(QVector<QLineEdit*>());
    for(int column=0;column<n;column++){
        array[row].append(new QLineEdit);
    }
}
for(int row=0;row<n;row++){
    for(int column=0;column<n;column++){
        array[row][column]->setFixedSize(40,40);
        array[row][column]->setText("");
        array[row][column]->setAlignment(Qt::AlignCenter);
        grid->addWidget(array[row][column],row,column);
    }
}
grid->setSizeConstraint(QLayout::SetMaximumSize);
QHBoxLayout *detStack = new QHBoxLayout();
QWidget *subMatrixFrame = new QWidget();
detStack->addWidget(subMatrixFrame);
subMatrixFrame->setLayout(grid);
matrixFrame->setLayout(detStack);

QWidget *result = new QWidget;
QHBoxLayout *resultLayout = new QHBoxLayout;
QLabel *resultLabel = new QLabel("");

result->setLayout(resultLayout);
resultLayout->addWidget(resultLabel);
determinantLayout->addWidget(pushButtonTop);
determinantLayout->addWidget(matrixFrame);
determinantLayout->addWidget(result);
```

Slika 3.15. Kreiranje prikaza matrice i njenih elemenata u grafičkom sučelju

QVector klasa nam omogućava dinamičko kreiranje polja, koja će u ovom slučaju biti korištena za prikaz i unos elemenata matrice. Sastoji se od QLineEdit tipova podataka koji će predstavljati svaki pojedini element u svom odgovarajućem retku i stupcu. QLineEdit tipovi podataka kreiraju mali prostor za unos teksta u koji mi možemo unijeti željene vrijednosti elemenata matrice.

Postavljamo veličinu polja koje služi za unos elemenata kao i tekst koji se nalazi unutar QLineEdita. U ovom slučaju među navodnicima nema ništa, što govori da će prostor unutar polja za unos teksta biti prazan. Centriramo tablicu QlineEditora i dodajemo kreirane *widgete* u pripadajući raspored.

Kreira se novi *widget* zvan subMatrixFrame koji će biti zadužen da nakon promjene veličine matrice, kreira okvir za novu matricu. Nakon svakog pritiska na tipku resize, briše se postojeća matrica kao i dvodimenzionalno polje u koju su se spremali podaci koje smo unijeli u prostor za elemente matrice. Matrica je opet stavljena u Grid raspored koji najbolje predstavlja izgled matrice.

```
this->connect(calculateButton,&QPushButton::clicked,
    [this, resultLabel](){
        calculateDeterminant(resultLabel);
    }
);

this->connect(resizeButton,&QPushButton::clicked,
    [this,size,subMatrixFrame](){
        resizeMatrix(size,subMatrixFrame);
    }
);
this->connect(exitButton,&QPushButton::clicked,
    [this](){
       close();
    }
);
this->connect(colorButton,&QPushButton::clicked,
    [this,Widget,Frame,determinant](){
       toDarkTheme(Widget,Frame,determinant);
    }
);
this->connect(colorButton2,&QPushButton::clicked,
    [this,Widget,Frame,determinant](){
       toFeritTheme(Widget,Frame,determinant);
    }
);
```

Slika 3.16. Povezivanje signala i slotova

Slot se poziva kao odgovor na emitirajući signal povezan s njim. *Slotovi* su C ++ funkcije koje se pozivaju kao odgovor na određeni signal, mogu se normalno nazivati, te je njihova jedina posebna značajka što se na njih mogu povezati signali. Signal se emitira kada se dogodi određeni događaj.

U ovom djelu koda definiran je postupak povezivanja signala i slotova. Pomoću ovih funkcija aplikacija može reagirati na „događaje“, odnosno korisnikove unose i zahtjeve.

Prva funkcija connect povezuje klik na tipku sa nazivom calculate**,** sa funkcijom calculateDeterminant i daje joj vrijednost koja se nalazila u predviđenom prostoru za tekst za vrijeme klika na tipku.

Druga funkcija connect povezuje klik na tipku sa nazivom resize sa funkcijom resizeMatrix koja prima novu dimenziju matrice u obliku cjelobrojnog broja koja se nalazila u prostoru za tekst.

Također prima subMatrixFrame tipa widget, koja omogućava da se nove dimenzije matrice prilagode u *grid* raspored.

Treća funkcija connect povezuje klik na tipku sa nazivom exit, te je povezuje sa jednostavnom funkcijom close koja zatvara aplikaciju.

Četvrta i peta funkcija connect odgovorne su za tematsku promjenu boja. Mijenjaju boju okvira kao i pozadinsku boju.

```
void MainWindow::toDarkTheme(QWidget *Widget,QWidget *Frame,QWidget *determinant){
    QPalette pal = palette();
    pal.setColor(QPalette::Background, Qt::darkGray);
    determinant->setPalette(pal);
    determinant->show();
    determinant->setAutoFillBackground(true);

    QPalette pal2 = palette();
    pal2.setColor(QPalette::Background, Qt::black);
    Frame->setPalette(pal2);
    Frame->show();
    Frame->setAutoFillBackground(true);

    QPalette pal3 = palette();
    pal3.setColor(QPalette::Background, Qt::black);
    Widget->setPalette(pal3);
    Widget->show();
    Widget->setAutoFillBackground(true);
}
void MainWindow::toFeritTheme(QWidget *Widget,QWidget *Frame,QWidget *determinant){
    QPalette pal = palette();
    pal.setColor(QPalette::Background, Qt::white);
    determinant->setPalette(pal);
    determinant->show();
    determinant->setAutoFillBackground(true);

    QPalette pal2 = palette();
    pal2.setColor(QPalette::Background, Qt::red);
    Frame->setPalette(pal2);
    Frame->show();
    Frame->setAutoFillBackground(true);

    QPalette pal3 = palette();
    pal3.setColor(QPalette::Background, Qt::blue);
    Widget->setPalette(pal3);
    Widget->show();
    Widget->setAutoFillBackground(true);
}
```

Slika 3.17. Funkcije za tematske promjene sučelja

```
void MainWindow::calculateDeterminant(QLabel *label){
    Matrix matrix(n);
    for(int row=0;row<n;row++){
        for(int column=0;column<n;column++){
            float value = array.at(row).at(column)->text().toFloat();
            matrix.setElement(row,column,value);
        }
    }
    float determinant_result = matrix.calculateRecursively();
    label->setText("Determinant of a matrix is:  " + QString::number(determinant_result));
    label->setAlignment(Qt::AlignCenter);

}

void MainWindow::resizeMatrix(QLineEdit* dimensions, QWidget *matrixFrame){
    int range = dimensions->text().toInt();
    if(range>10){
        range=3;
}
    for(int row=0;row<n;row++){
        for(int column=0;column<n;column++){
            delete array[row][column];
        }
    }
    array.clear();
    n = range;
    delete grid;
    grid = new QGridLayout();
    grid->setSizeConstraint(QLayout::SetMaximumSize);
    for(int row=0;row<n;row++){
        array.append(QVector<QLineEdit*>());
        for(int column=0;column<n;column++){
            array[row].append(new QLineEdit);
        }
    }
    for(int row=0;row<n;row++){
        for(int column=0;column<n;column++){
            array[row][column]->setFixedSize(40,40);
            array[row][column]->setText("");
            array[row][column]->setAlignment(Qt::AlignCenter);
            grid->addWidget(array[row][column],row,column);
        }
    }
    matrixFrame->setLayout(grid);
    grid->update();
    matrixFrame->update();
}
```

```
MainWindow::~MainWindow(){
    delete gui;
}
```

Slika 3.18. Funkcije grafičkog sučelja

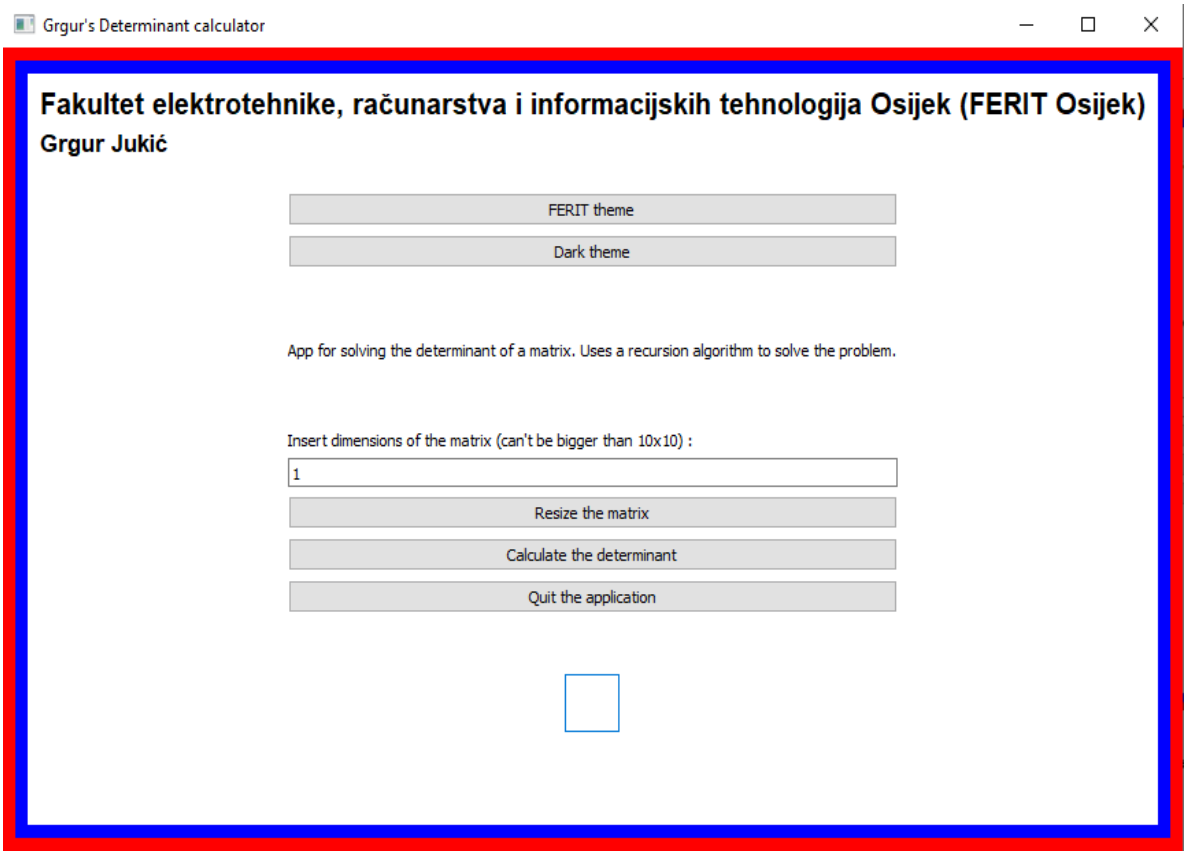
CalculateDeterminant prima vrijednost tipa QLabel (polja u koje se unosi vrijednost elementa matrice). Kreira matricu matrix veličine n, te popunjava polje sa vrijednostima koje su unesene u prostor za tekst na grafičkom sučelju. Pretvaramo brojeve koje smo unijeli kao *stringove* u prave cjelobrojne vrijednosti. Postavljamo elemente te pozivamo metodu za izračun determinante. Grafičko sučelje prikazuje rezultat izračuna determinante matrice kao label. Centriramo rezultat pomoću AlignCenter.

resizeMatrix prima vrijednost tipa QLineEdit, tekst editor u kojem se nalazi vrijednost koju je korisnik unio kao željenu vrijednost dimenzija matrice. Ovdje je postavljeno ograničenje, da ukoliko je korisnik unio cjelobrojnu vrijednost veću od 10 postavlja se veličina matrice 3x3. Briše se trenutno polje kao i *grid layout*. Nakon toga, kreira se nova matrica na grafičkom sučelju, spremna za unos elemenata matrice. Funkcija update() ažurira layout prema veličini nove matrice i njenog grid layouta.

~MainWindow je klasični C++ destruktor.

4. GRAFIČKO SUČELJE APLIKACIJE

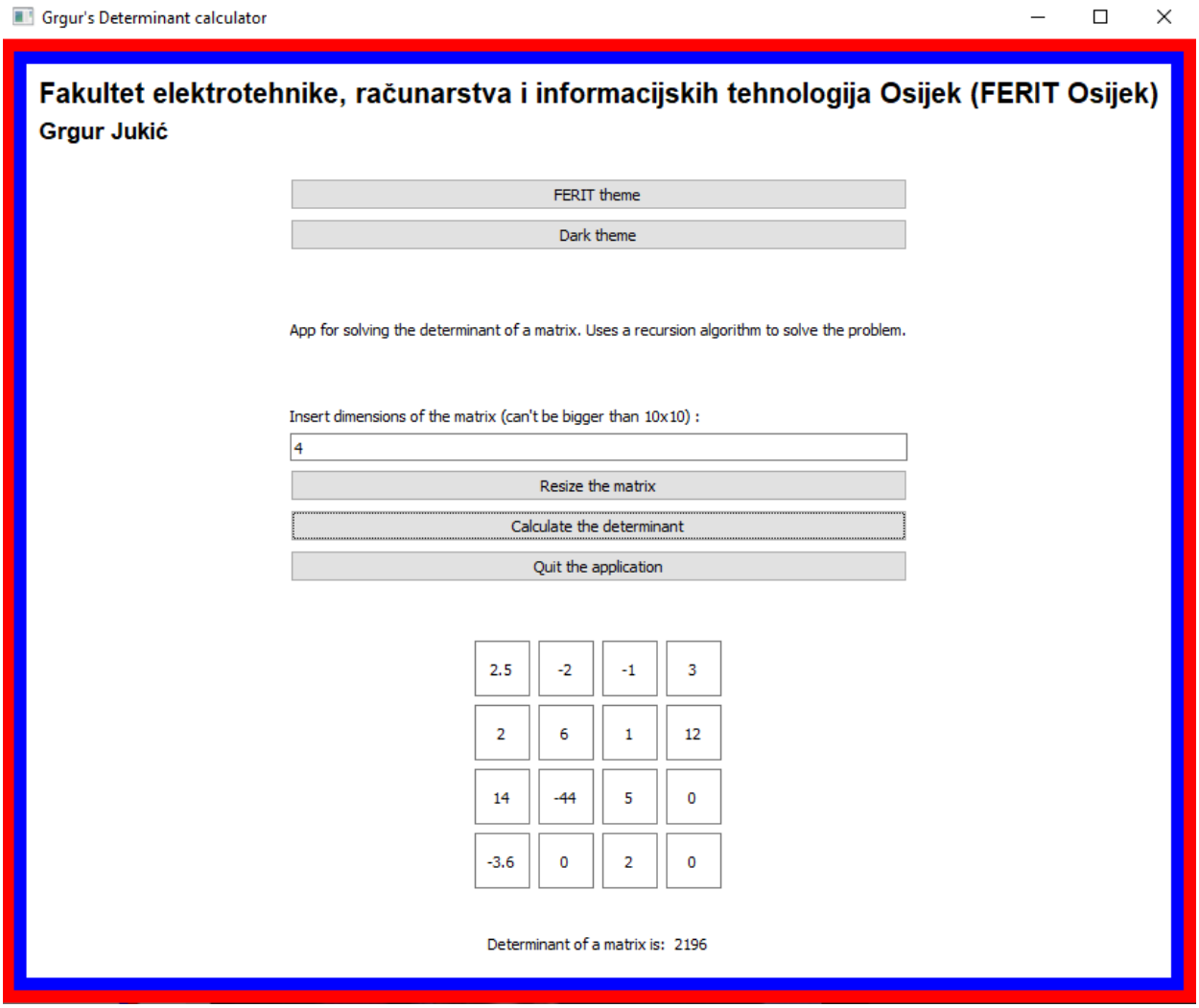
4.1. Prikaz sučelja pri pokretanju aplikacije



Slika 4.1. Prikaz sučelja prilikom prvog pokretanja aplikacije

Sučelje se sastoji od tekst labela koji nam govori čemu služi line editor. Unosimo veličinu matrice koju želimo te popunjavamo ćelije koje predstavljaju elemente matrice. Nakon unosa elemenata pritisnemo tipku za izračun determinante, te nam se rezultat matrice ispiše na dnu prozora.

4.2. Prikaz sučelja i izračuna determinante za 4x4 matricu

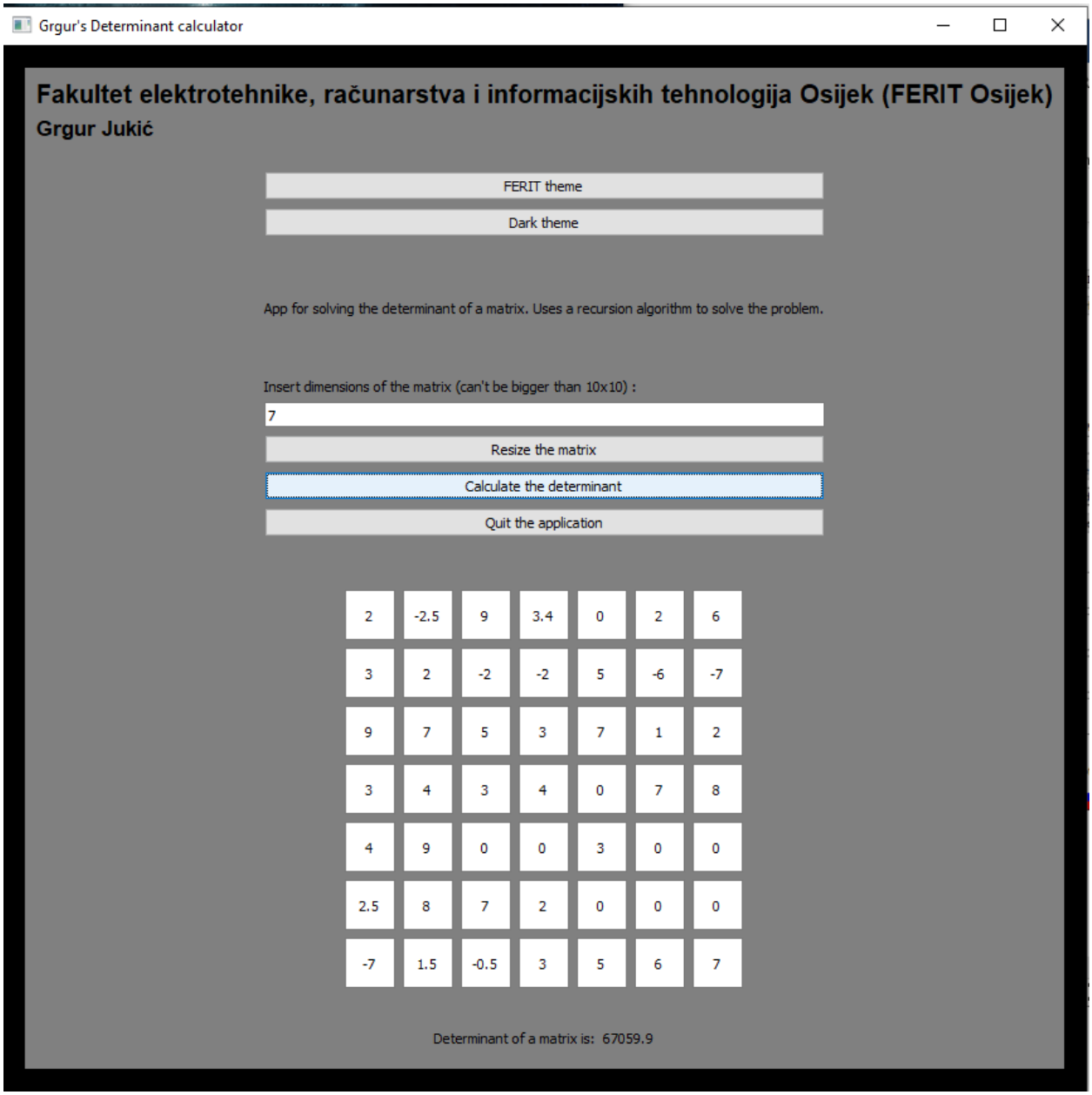


Slika 4.2. Prikaz izračuna determinante 4x4 matrice

U ovom primjeru, želimo izračun matrice 4x4. Unosimo u line editor brojku 4 koja predstavlja dimenzije kvadratne matrice. Nakon toga naša matrica se proširuje i prikazuje sve ćelije koje predstavljaju elemente matrice.

Unosimo elemente u prostor za tekst, te nakon pritiska na tipku Calculate, u donjem djelu sučelja dobijemo rješenje determinante

4.3. Prikaz sučelja i izračuna determinante za 7x7 matricu, tamna tema



Slika 4.3. Prikaz izračuna determinante matrice 7x7 koristeći tamnu temu

Klikom na tipku sa nazivom „Dark theme“ promjeni se boja okvira u čvrstu crnu, dok se boja pozadine promjeni u tamno sivu.

5. ZAKLJUČAK

Razvoj i proširenje znanja, sakupljanje intelektualnog kapitala, moraju pratiti razvoj novih tehnologija i skupa alata području računarstva i informacijskih znanosti. Pošto je područje znanosti koje mi proučavamo podložno ubrzanim promjenama bitno je ostati u toku, učiti i proučavati nove tehnologije i skupove alata koji nam postaju dostupnima svakim danom. Skoro svake godine, dobivamo na pristup nove operativne sustave, nove programske jezike, proširenja za već postojeće jezike, te je na nama da gradimo na znanju koje smo dosad prikupili.

Aplikacija radi upravo to, upotrebljava se znanje koje je pruženo na fakultetu, uz znanja koja su se morala samostalno prikupiti. Pruža korisnicima da na brz način, uz upotrebu laganog korisničkog sučelja, izračunaju determinantu matrice proizvoljne veličine. Koristeći Qt, omogućeno je pravljenje grafičkog sučelja kao i mogućnost korištenja aplikacije na više različitih platforma. Kroz izradu završnog rada, upoznao sam se sa Qt bibliotekama i svim mogućnostima Qt frameworka. Naučio sam princip signala i slotova koji imitiraju C# funkcionalnost događaja. Ugradnjom sučelja pomoću Qt biblioteka pokazao sam nekakvu dodatnu nišu C++ jezika, maknuvši se od klasičnog unosa i ispisa iz konzole.

Aplikacija ima još puno mjesta za napredak. Vjerujem da se kod mogao napisati efikasnije i jednostavnije, no to se može riješiti dodatnim refaktoriranjima kroz vrijeme. Sama aplikacija, kako bi računala determinantu matrice, već ima velik broj metoda koje su potrebne i za neke druge matematičke radnje sa matricom. Moguće je lagano dodati funkcionalnost koja prikazuje transponiranu matricu ili potenciranje matrica, množeći matricu samu sa sobom.

LITERATURA

[1] Linearna Algebra, Damir Bakić
https://web.math.pmf.unizg.hr/nastava/la1/dodatno/2019_LA_drugo_izdanje_v6.pdf (datum posljednje posjete: 13.9.2020.)

[2] Geeksforgeeks, LU dekompozicija
https://www.geeksforgeeks.org/l-u-decomposition-system-linear-equations/ (datum posljednje posjete: 19.6.2020.)

[3] Math UCLA, QR dekompozicija, Gram Schmidt
https://www.math.ucla.edu/~yanovsky/Teaching/Math151B/handouts/GramSchmidt.pdf (datum posljednje posjete 19.6.2020.)

[4] W3schools, Što je C++
https://www.w3schools.com/cpp/cpp_intro.asp(datum posljednje posjete: 19.6.2020.)

[5] Tutorialspoint, C++ Nasljeđivanje
https://www.tutorialspoint.com/cplusplus/cpp_inheritance.htm (datum posljednje posjete: 10.7.2020.)

[6] Geeksforgeeks, Polimorfizam u C++ jeziku
https://www.geeksforgeeks.org/polymorphism-in-c/ (datum posljednje posjete: 10.7.2020.)

[7] Tutorialspoint, Enkapsulacija podataka u C++https://www.tutorialspoint.com/cplusplus/cpp_data_encapsulation.htm (datum posljednje posjete: 12.7.2020.)

[8] Geeksforgeeks, Apstrakcija u C++ jeziku
https://www.geeksforgeeks.org/abstraction-in-c/ (datum posljednje posjete: 12.7.2020.)

[9] Qt.io, Informacije o Qtu

https://wiki.qt.io/About_Qt (datum posljednje posjete: 13.7.2020.)

[10] Qt.io, Informacije o tvrtki

https://wiki.qt.io/About_Qt (datum posljednje posjete: 13.7.2020.)

SAŽETAK

Izrada C++ aplikacije za izračun determinante kvadratne matrice, uključujući izradu korisničkog grafičkog sučelja u Qt Creatoru. Omogućiti korisniku da kroz jednostavno sučelje izabere veličinu matrice, te nakon unosa njenih elemenata, izračuna determinantu matrice. Prvi dio rada govori o alatima korištenim za izradu završnog rada uključujući programski jezik C++, Qt *framework* i Qt skup alata za izradu aplikacije. Drugi dio rada govori o samoj strukturi aplikacije, izvornim datotekama i datotekama zaglavlja kao i o datoteci za kreiranje grafičkog sučelja. U trećem djelu prikazujemo dizajn grafičkog sučelja koji uključuje glavni prozor, ćelije koje predstavljaju elemente matrice, dvije tipke za promjenu veličine matrice, odnosno za izračun determinante matrice.

Ključne riječi:

C++, determinanta, grafičko sučelje, matrice, Qt

SUMMARY

Development of a C ++ application for calculating the determinant of a square matrix, including the development of a graphical user interface in Qt Creator. Allows the user to select the size of the matrix through a simple interface, and after entering its elements, calculates the determinant of the matrix. The first part of the paper discusses the tools used to create the final application including the C ++ programming language, the Qt framework, and the Qt set of application development tools. The second part of the paper talks about the structure of the application, source files and header files, as well as the file for creating a graphical interface. In the third part, we show the design of the graphical interface that includes the main window, cells that represent the elements of the matrix, two buttons to change the size of the matrix, or to calculate the determinant of the matrix.

Key words:

C++, determinant, matrix, user interface, Qt

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

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

Osijek, 17.09.2020.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Grgur Jukić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3647, 16.09.2019.
OIB studenta:	10983666859
Mentor:	Doc.dr.sc. Tomislav Rudec
Sumentor:	Izv. prof. dr. sc. Alfonzo Baumgartner
Sumentor iz tvrtke:	
Naslov završnog rada:	Rješavanje determinante u programskom jeziku C++
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 2 razina
Datum prijedloga ocjene mentora:	17.09.2020.
Datum potvrde ocjene Odbora:	

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 23.09.2020.

Ime i prezime studenta:	Grgur Jukić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3647, 16.09.2019.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Rješavanje determinante 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: