Računanje ranga matrice u programskom jeziku C

Pandurić, Iva

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

RAČUNANJE RANGA MATRICE U PROGRAMSKOM JEZIKU C

Diplomski rad

Iva Pandurić

Osijek, 2018.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 07.09.2018.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Iva Pandurić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 875 R, 24.09.2017.
OIB studenta:	31211929461
Mentor:	Doc.dr.sc. Tomislav Rudec
Sumentor:	Doc.dr.sc. Alfonzo Baumgartner
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc.dr.sc. Tomislav Keser
Član Povjerenstva:	Doc.dr.sc. Alfonzo Baumgartner
Naslov diplomskog rada:	Računanje ranga matrice u programskom jeziku C
Znanstvena grana rada:	**Procesno računarstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Student će izraditi aplikaciju u programskom jeziku C koja će za zadanu matricu ispisati njen rang.
Prijedlog ocjene pismenog dijela ispita (diplomskog 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 mentora:	07.09.2018.

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, 10.09.2018.

Ime i prezime studenta:	Iva Pandurić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 875 R, 24.09.2017.
Ephorus podudaranje [%]:	9%

Ovom izjavom izjavljujem da je rad pod nazivom**: Računanje ranga matrice u programskom jeziku C**

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

i sumentora Doc.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 ovog diplomskog rada je izraditi program u programskom jeziku C koji izračunava rang unešene matrice. Glavni cilj rada je omogućiti jednostavno određivanje ranga matrice pomoću računalnog programa, bez primjene vlastitog znanja linearne algebre. Kako ovakav zadatak ne zahtjeva posebno grafičko sučelje, nego je dovoljno samo unijeti potrebne podatke, program je napisan u programskom jeziku C. Osnove jezika sežu u područje matematike, pa je samim time i najpogodniji za izradu programa matematičke tematike.

Kako bi uspjela što kvalitetnije napraviti ovaj program morala sam se dobro upoznati sa programskim jezikom C i sa gradivom kolegija Linearna algebra, posebno onim koji se odnosi na matrice. Za rad sa programskim jezikom C na Linux platformi nije potrebno nikakvo razvojno okruženje, dok je na operacijskom sustavu Windows korišten Microsoftov Visual Studio.

Prvi dio diplomskog rada sadržava teorijsku podlogu, odnosno detaljan opis programskog jezika C i razvojnog okruženja koji su korišteni pri izradi programskog rješenja. Nadalje, opisana je linearna algebra, posebice dio algebre koji se odnosi na matrice. Nakon toga je opisano programsko rješenje, korak po korak, objašnjeno s matematičke strane i potkrijepljeno dijelovima koda.

1.1. Zadatak diplomskog rada

Zadatak diplomskog rada je izraditi program, u programskom jeziku C, koji će računati rang zadane mu matrice koristeći algoritam za računanje.

2. TEORIJSKE PODLOGE

2.1. Programski jezik C

C je programski jezik opće namjene kojeg je sedamdesetih godina prošlog stoljeća kreirao Dennis Ritchie. Jezik je prvotno korišten u svrhu kodiranja sistemskih programa i jezgre operacijskog sustava UNIX. Ubrzo se jezik počeo razvijati i sve se više koristio, ne samo za sistemsko programiranje, nego u brojne druge svrhe.

Kao podlogom za kreiranje jezika Richie se poslužio do tada poznatim jezikom B koji je napisao Ken Thompson 1970. godine, a preko B jezika ostvarena je i poveznica sa Richardsovim BCPL jezikom. Oba ova jezika ne sadrže tipove podataka, što je prva velika razlika koja je uvedena nastankom C jezika. Jezik se dosta mijenjao tijekom godina i prva važnija verzija „K&R C“ izdana je 1978. godine te je ona postala standard za jezik u sedamdesetim godinama i dugi je niz godina služila kao jedina definicija jezika. Pet godina kasnije, 1983. ANSI osniva udrugu koja brine o cjelokupnoj i modernoj definiciji C-a i taj, ANSI C standard, je usvojen 1988. godine. Do danas je C ostao vrlo bitan i rasprostranjen programski jezik prvenstveno zbog svoje prilagodljivosti svim platformama, od manjih sustava sve do superračunala. C koristi iste vrste objekata koje koristi i samo računalo, to su znakovi, adrese i brojevi, a njima se upravlja pomoću aritmetičkih i logičkih operatora. Za dobro razumjevanje i upotrebu C-a poželjno je poznavanje načina na koji radi računalo, odnosno procesor, memorija, ulazno-izlazni sklopovi, itd. C je nezavisan od arhitekture sustava i moguće je napisati prenosive programe koji se mogu koristiti bez potrebe za mijenjanjem sklopovlja. Jezik sam po sebi ne radi direktno sa nekim složenijim objektima, no za to postoje biblioteke koje nužno prate C jezik, a koje su doprinos ANSI C standarda. Biblioteke osiguravaju pristup operativnom sistemu, formiraju ulaz i izlaz, određuju položaj u memoriji, rade s nizovima, itd. Više o bibliotekama, kao i o ostalim obilježjima programskog jezika rečeno je u idućim poglavljima.

Kao i svaki drugi programski jezik i C ima poneke nedostatke. Kao najveći nedostatak navodi se problem upravljanja memorijom jer prilikom pisanja koda programer mora sam alocirati potrebne memorijske resurse što često može stvoriti problem, osobito neiskusnijim programerima. Baš iz tog razloga većina programera koristi druge jezike ukoliko nije nužno koristiti C te se njegova upotreba svodi na sistemske programe, programe prevoditelje i jezgre operativnih sustava, iako su mu mogućnosti puno veće. [1, 2]

2.1.1. Tipovi i operatori

Osnovni oblici podataka s kojima se radi u programu su varijable i konstante. Varijable se na početku programa deklariraju, tj. definira se popis varijabli koje će se koristiti, njihov tip i ako je potrebno, početna vrijednost. Tip definira skup vrijednosti koje neki objekt može poprimiti i operacije koje se nad njim mogu izvršavati. Kombinacijom varijabli i konstanti dobivamo izraze kojima proizvodimo nove vrijednosti.

U C-u se koristi nekoliko osnovnih tipova podataka [3]:

- int – cjelobrojna vrijednost, raspon vrijednosti od -32,768 do 32,767
- char – jedan znak iz skupa znakova, raspon vrijednosti od -128 do 127
- float – realni broj s jednostrukom preciznošću, raspon vrijednosti od 1.2E-38 do 3.4E+38
- double – realni broj s dvostrukom preciznošću, raspon vrijednosti od 2.8E-308 do 1.7E+308

Tipovima podataka, ukoliko je potrebno, pridružuju se kvantifikatori. Rasprostranjeni su kvantifikatori long, short, signed i unsigned. Oni se koriste zajedno s tipovima podataka i mijenjaju raspon vrijednosti osnovnom tipu s kojim se koriste.

Prilikom pisanja programa u C-u koriste se konstante, odnosno neke fiksne vrijednosti koje se ne mijenjaju tijekom izvođenja programa. One mogu biti cjelobrojne i znakovne. Uvodimo ih u program pomoću pretprocesorske naredbe #define ili ključne riječi const. U kontekstu znakovnih konstanti često se koriste escape nizovi, odnosno dva znaka koja predstavljaju jedan i imaju neku funkciju, kao na primjer '\n' što označa novi red. [2]

Programski jezik C koristi različite vrste operatora. Binarni aritmetički operatori su +, -. *, / i modul %, tj. ostatak cjelobrojnog dijeljenja. Operatori *, / i % imaju viši prioritet od binarnih operatora + i – koji su istog prioriteta. Relacijski operatori su <, <=, >, >= i svi oni imaju isti prioritet, ali niži od aritmetičkih operatora. Po prioritetu su odmah ispod njih operatori jednakosti == i !=. Logički su pak operatori &&, || i !, tj. logički i, ili i ne. C ima i operatore inkrementiranja (uvećavanja) ++ koji dodaje jedan svom operandu, te operator dekrementiranja (umanivanja) -- koji oduzima jedan svom operandu. Oba se mogu koristiti i kao prefiks i kao sufiks operatori te je potrebno pripaziti prilikom njihova korištenja. Postoje još i operatori dodjeljivanja vrijednosti, od kojih su najkorišteniji =, += (npr. a += b ekvivalentno a = a + b), -=, *=, /=, %=. Važni su još operator sizeof() koji vraća veličinu varijable, & koji vraća adresu varijable, * što označava pokazivač i ternarni operator ? : koji se korisit u uvjetnom izrazu. [3]

2.1.2. Kontrola toka

Za kontrolu toka programa, odnosno redoslijed izvršavanja operacija, bitni su grananje i petlje. Grananje se odvija pomoću naredbi. Neki izraz u C-u postaje naredba kada je popraćen točka-zarezom (';') i svaka naredba nužno završava tim znakom. Za grupe deklaracija i naredbi koriste se vitičaste zagrade {}.

Naredbe koje se koriste u C-u su if, if - else i switch - case. If naredba se sastoji od boolean, odnosno logičkog izraza, koji je ili točan ili netočan i sukladno tome se izvodi sljedeći kod. If – else naredba je slična prethodnoj samo što sadrži else dio koji se izvodi ukoliko izraz unutar if naredbe nije istinit. Ukoliko je potrebno provjeriti više uvjeta za redom, iza if naredbe koristimo naredbu else if, te tako pišemo svaku sljedeću naredbu do poslijednje else naredbe.

```
if (izraz)
      naredba
else if (izraz)
      naredba
else
       naredba
```

Switch - case naredba predstavlja višesmjerno grananje koje provjerava odgovara li izlaz nekoj od konstanti cjelobrojnih vrijednosti i na osnovu toga se grana. Ukoliko uvjet odgovara nekoj od vrijednosti izraza, obavljaju se naredbe za taj slučaj, a pri tome svi izrazi moraju biti različiti. Postavlja se jedan podrazumijevani, *default* izlaz koji se obavlja ukoliko nijedan uvjet nije ispunjen. Trenutni izlaz iz switch petlje obavlja se pomoću naredbe break. [5]

```
switch(uvjet)
{
    case (konstantni_izraz1): blok naredbi
    break;
    case (konstantni_izraz2): blok naredbi
    break;
    default: blok naredbi
}
```

Vrlo bitan faktor prilikom kontrole toka programa su petlje. U programskom jeziku C najčešće se koriste tri petlje: for, while i do – while. Petlje se, kao i naredbe, mogu koristiti ugnježđeno, jedna unutar druge. For petlja se najčešće koristi ako se blokovi naredbi ponavljaju unaprijed poznati

broj puta. Unutar petlje zadamo početno stanje, uvjet i prirast, tj. za koliko se mijenja iznost vrijednosti kontrolne varijable koja je postavljena u uvjetu.

```
for (pocetno_stanje; uvjet; prirast)
{
      blok naredbi
}
```

Za razliku od for petlje, while petlju koristimo kada ne znamo unaprijed koliko će se puta petlja ponoviti, nego to ovisi o ispunjenju zadanog uvjeta. Dok je neki uvjet istinit ispunjava se blok naredbi. Uvjet se ispituje na početku i može se dogoditi da se blok naredbi unutar petlje uopće ne izvrši.

```
while (uvjet)
{
      blok naredbi
}
```

Upravo se po tome do – while petlja razlikuje od while petlje. Uvjet se ispituje tek nakon izvođenja bloka naredbi, a blok se mora izvršiti jednom i izvršava se sve dok je uvjet točan.

```
do
{
      blok naredbi
}
while (uvjet);
```

Ukoliko želimo mogućnost izlaska iz petlje, to možemo postići koristeći naredbu break. Program nailaskom na break trenutno prekida izvođenje petlje koja se izvodila do tog trenutka. S njom je povezana i naredba continue čiji je zadatak da započne sljedeću iteraciju petlje u kojoj se program vrti, npr. kod for petlje trenutno se prelazi na stupanj prirasta, a kod do – while i while na provjeru uvjeta. [2, 3]

2.1.3. Funkcije

Ukoliko se neki dijelovi koda višestruko ponavljaju moguće ih je pohraniti u obliku funkcija. Funkcije su grupe naredbi koje zajedno izvršavaju neki zadatak i moguće ih je više puta koristiti, odnosno pozivati. Svaki program u C-u obavezno posjeduje glavnu main() funkciju. Programer

može sam napisati, odnosno deklarirati svoju funkciju, a može korisitit i prethodno napisane funkcije. Prilikom deklariranja funkcije potrebno je navesti njezino ime, tip podatka koji funkcija vraća i argumente koji joj se predaju. Unutar tijela funkcije, u vitičastim zagradama, nalazi se definicija funkcije i povratna vrijednost, ukoliko postoji. Definicija se sastoji od naredbi koje obavljaju operacije nad predanim parametrima.

```
povratni_tip naziv_funkcije (tip arg1, tip arg2)
{
      popis naredbi;
      return vrijednost;
}
```

Funkcije se pišu svaka za sebe, izvan glavne main() funkcije i pozivaju se kada su potrebne, najčešće unutar main() funkcije. Poziv se vrši tako da se navede ime funkcije i predaju joj se argumenti, onim redosljedom kako su navedeni u deklaraciji funkcije. Ukoliko funkcija ne vraća nikakvu vrijednost, ona je tipa void i unutar tijela funkcije nije potrebno pisati ključnu riječ return. Prilikom poziva funkcije postoje dva načina na koji argumenti mogu biti prosljeđeni funkciji, po vrijednosti i po referenci. Ako prosljeđujemo po vrijednosti argument vrijedi samo u funkciji i njegova vrijednost se na izlasku iz funkcije gubi, tj. promjena njegove vrijednosti unutar funkcije ne utječe na stvarni argument s kojim smo funkciju pozvali, a prilikom prosljeđivanja po referenci koristimo pokazivače i umjesto stvarnih vrijednosti prosljeđuju se adrese argumenata i mijenja se vrijednost stvarnim argumentima. U C-u se još često koriste funkcije koje pozivaju same sebe, tzv. rekurzivne funkcije. [2, 3]

2.1.4. Polja i pokazivači

Polja sadrže podatke istog tipa koji su poredani u niz, jedan iza drugoga, a svaki podatak posjeduje cjelobrojni indeks pomoću kojeg mu se može pristupiti, a koji označava udaljenost podatka od prvog člana polja. Prvi član polja ima indeks 0, a posljednji N-1, gdje je N veličina polja. Prije korištenja polje je potrebno deklarirati i odrediti mu maksimalnu duljinu. Duljina nužno mora biti veća ili jednaka broju članova polja.

```
tip_podataka ime_polja [veličina];
```

Ukoliko programer sam želi inicijalizirati polje, vrijednosti unosi u vitičaste zagrade i odvaja ih zarezom.

```
int ime_polja [3] = {1, 2, 3};
```

Takva polja nazivaju se jednodimenzionalna polja. Uz njih, postoje još i višedimenzionalna polja. Takva polja možemo zamisliti kao tablicu koja ima retke i stupce gdje prvi indeks određuje redak dok drugi indeks određuje stupac. Deklariranje se vrši isto kao i kod jednodimenzionalnih polja, dok se inicijalizacija vrši na način da se unutar vanjskih vitičastih zagrada dva puta unose vrijednosti u unutarnje vitičaste zagrade međusobno odvojene zarezom.

Ukoliko želimo unijeti neki znakovni niz to je također moguće pomoću polja. Potrebno je napraviti polje tipa char. Svaki takav niz završava sa ključnim znakom '\0' koji zauzima jedno mjesto u memoriji.

Vrlo važan alat u C-u su pokazivači koji omogućavaju lakše izvođenje nekih operacija i bez kojih dinamičko alociranje memorije ne bi bilo moguće. Svaka varijabla je prilikom deklaracije spremljena na neku lokaciju u memoriji. Memoriju možemo zamisliti kao jednodimenzionalno polje *byteova* gdje svaki *byte* ima adresu. Upravo tim adresama se pristupa pomoću pokazivača. Laički rečeno, pokazivač je varijabla koja pokazuje na memorijsku adresu neke druge varijable. Deklarira se tako da se navede tip u ovisnosti na koji tip varijable se pokazivač odnosi i ime varijable ispred kojeg se stavlja *.

```
tip *ime_varijable;
```

Adresu od neke varijable saznajemo koristeći operator &. Recimo da želimo saznati adresu varijable a i spremiti ju u pokazivač p, to ćemo učiniti na sljedeći način: [4]

```
int a;
int *p;
p = &a;
```

Pokazivači i polja su vrlo tijesno povezani. Svaka operacija koja se može napraviti pomoću indeksa polja, može se također napraviti i koristeći pokazivače. Pomoću pointerske aritmetike, koja je dio C standarda, polje identificiramo s pokazivačem na prvi element polja. Pokazivači su i sami varijable stoga se mogu pohraniti u poljima kao i sve druge varijable pomoću pokazivača na pokazivače. Pokazivači su jako korisni, ali njihovo korištenje može biti opasno ukoliko se koriste nepravilno. [2, 3]

2.1.5. Strukture i unije

Struktura je skup jedne ili više varijabli, istog ili različitog tipa, koje su grupirane zajedno. One nam daju mogućnost kreiranja „novih tipova“ praveći kolekciju postojećih tipova. Za definiranje strukture koristi se ključna riječ struct.

```
struct struktura {
            tip_podatka ime1;
            tip_podatka ime2;
              . . .
};
```

Varijable tipa struktura deklariramo kao `struct struktura sIme`, a pokazivače kao `struct struktura* pIme`. Članovima strukture pristupamo pomoću operatora '.' koji se nalazi između imena strukture i imena člana strukture kojem želimo pristupiti, npr. `sIme.ime1` ili preko pokazivača `pIme->ime1`. [5]

Unija je varijabla koja omogućuje spremanje objekata različitih tipova i veličina na istu memorijsku lokaciju u različitim trenucima. Definiraju se slično kao strukture.

```
union unija {
     tip_podatka ime1;
     tip_podatka ime2;
       . . .
} u;
```

Varijabla u prima svaki od navedenih tipova i dovoljno je velika da primi najveći od njih, a veličina ovisi o implementaciji. Svaki od tipova se sprema u istu u varijablu i koristi se najsvježije pohranjeni tip, osim ako u programu nije napisano drugačije o čemu brigu vodi programer. [3]

2.1.6. Ulaz, izlaz i pretprocesorske naredbe

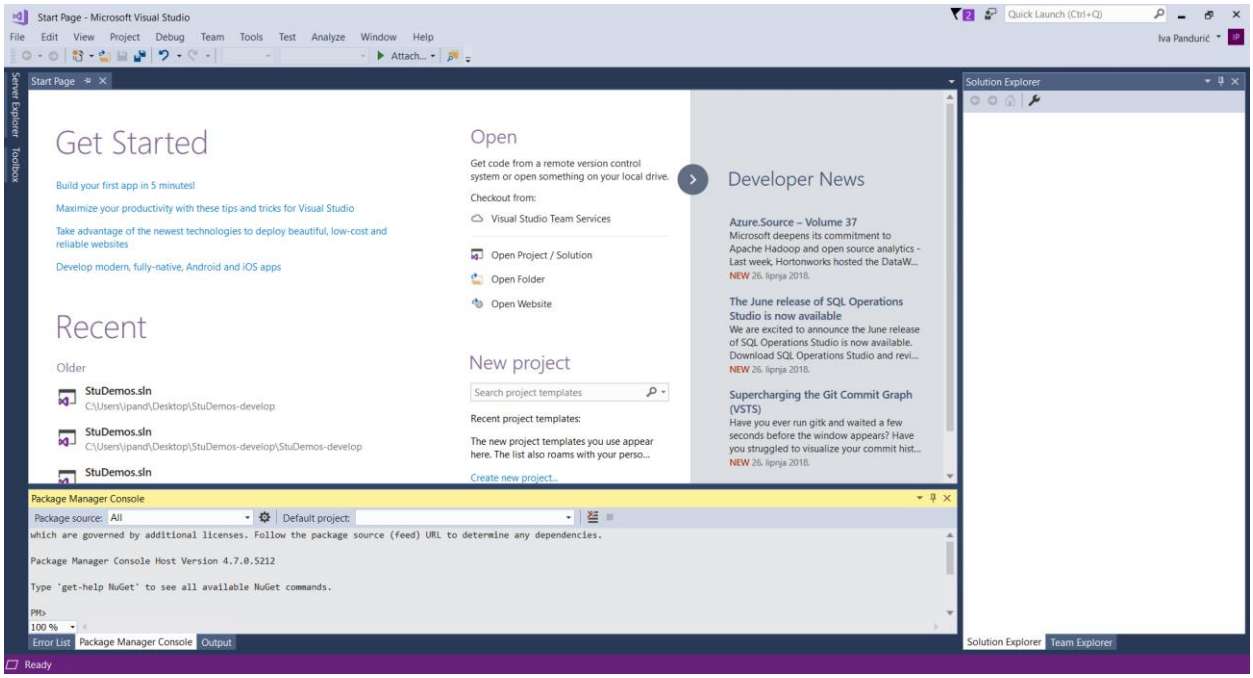
Samom jeziku C nedostaju neke važne mogućnosti i iz tog razloga koristi ranije spomenute biblioteke. C koristi skup funkcija koje određuju ulaz i izlaz, manipulaciju nizovima i memorijom, matematičke rutine kao i veliki broj drugih servisa za C programe, a koje su dio najkorištenije, standardne biblioteke definirane od strane ANSI standarda. Osobine funkcija ove biblioteke određene su u nekoliko zaglavlja od kojih je najpoznatija svakako datoteka zaglavlja stdio.h koja

sadrži deklaracije funkcija za ulaz i izlaz podataka, kako i samo ime „*standard input/output header*“ sugerira. Biblioteka sadrži brojne funkcije od kojih su najkorištenije printf za ispis podataka na standardni izlaz i scanf za učitavanje podataka sa standardnog ulaza. Prilikom ispisa i upisa potrebno je paziti na osnovne pretvorbe, odnosno pomoću kojih se znakova ispisuju i upisuju argumenti, npr. %d za varijablu tipa int. [6]

Biblioteke se u program uvode koristeći pretprocesorske naredbe, odnosno naredbe koje se izvršavaju prije početka izvršavanja izvornog koda programa. C pretprocesor nije dio kompajlera, računalnog programa koji prevodi kod napisan u određenom programskom jeziku u ciljani jezik nižeg nivoa, nego je dio u procesu prevođenja, odnosno samo navodi prevodilac da obavi naredbe prije stvarnog prevođenja. [7] Sve pretprocesorske naredbe započinju znakom #. Dvije najkorištenije pretprocesorske naredbe su #include kojom uključujemo zaglavlje iz druge datoteke u naš program i upravo pomoću ove naredbe se koriste zaglavlja iz raznih biblioteka te #define koja služi za definiranje simboličkog imena ili simboličke konstante. [2]

2.2. Microsoft Visual Studio

Microsoft Visual Studio je Microsoftovo razvojno okruženje koje se koristi za razvoj računalnih programa, mobilnih i web aplikacija, web servisa i web stranica. Prva verzija ovog razvojnog okruženja Visual Studio 97 izašla je 1997. godine i od tada je izašlo još deset novih inačica, a trenutno se koristi Visual Studio 2017. Visual Studio uključuje *debugger*, računalni program za uklanjanje grešaka [8], koji radi na razini izvornog koda i kao *debugger* na strojnoj razini. Uz njega okruženje sadrži još različitih dodatka koji pomažu pri dizajniranju grafičkog sučelja, no u ovom radu takvi dodaci neće biti potrebni. Visual Studio podržava trideset šest različitih programskih jezika, a ugrađeni jezici su C, C++/CLI, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML i CSS. Ostali jezici dostupni su korištenjem dodataka, eng. *plug-ins*. [9]



Sl. 2.1. Izgled razvojnog okruženja Microsoft Visual Studio

2.3. Linearna algebra

Linearna algebra je matematička disciplina koja proučava vektorske prostore, linearne operatore, sustave linearnih jednadžbi i matrice. Konkretnu realizaciju ove grane matematike nalazimo u analitičkoj geometriji. Njezina vrijednost leži u primjenjivosti na razne discipline kao što su matematička fizika, apstraktna algebra, računarstvo, pa čak i ekonomija. U svakoj disciplini gdje se dani problem može aproksimirati nekim linearnih problemom linearna algebra dolazi do izražaja i nameće se kao logično sredstvo za rješavanje problema. Kako se u ovom radu rješava problem računanja ranga matrica, daljnji opis algebre prvenstveno će se odnositi na opis matrica kao jedne od algebarskih struktura. [10, 11]

2.3.1. Matrice

Recimo da su *m* i *n* prirodni brojevi. Matrica je pravokutna shema u *m* redaka i *n* stupaca oblika:

$$A = [a_{ij}] = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

Element matrice A označavamo sa a_{ij} ili $[A]_{ij}$.

Skup svih matrica tipa (m, n) označavamo sa $\mathbf{M}_{mn}$, a ako je $m = n$ riječ je o skupu kvadratnih matrica $\mathbf{M}_n$.

Nul matrica O je matrica kojoj su svi elementi jednaki 0.

Jedinična matrica $I \in \mathbf{M}_n$ je kvadratna matrica koja se zapisuje pomoću Kroneckerovog simbola $\delta_{ij} = \begin{cases} 1, ako\ je\ i = j \\ 0, \ ako\ je\ i \neq j \end{cases}$, odnosno ovakvoj matrici su elementi na glavnoj dijagonali jedinice, dok su svi ostali elementi nule. Glavna dijagonala matrice A sadrži elemente a_{11}, a_{22}, a_{33}, itd.

Matrica je dijagonalna ako je kvadratna i ako svi elementi izvan glavne dijagonale iščezavaju. Nadalje, matrica je gornjetrokutasta ili donjetrokutasta ako svi elementi ispod, odnosno iznad glavne dijagonale iščezavaju, odnosno ako su jednaki nuli. Matrica još može biti simetrična ako vrijedi $a_{ij} = a_{ji}$ i antisimetrična ako vrijedi $a_{ij} = -a_{ji}$. Ukoliko matrici A zamijenimo retke i stupce dobivamo transponiranu matricu i označavamo ju sa A^T. Za simetrične matrice vrijedi $A^T = A$, a za antisimetrične $A^T = -A$. Dvije matrice su jednake ako su istog tipa i ako su im svi odgovarajući

elementi jednaki. Odnosno, matrica $A \in \mathbf{M}_{mn}$ jednaka je matrici $B \in \mathbf{M}_{pq}$ ako vrijedi $m = p$, $n = q$ i $a_{ij} = b_{ij}$, za sve $1 \leq i \leq m$, $1 \leq j \leq n$. [13]

2.3.2. Operacije s matricama

Tri su glavne operacije s matricama:

- Zbrajanje matrica

  Zbroj $A + B$ matrica $A,B \in \mathbf{M}_{mn}$ je matrica $C \in \mathbf{M}_{mn}$ s elementima $c_{ij} = a_{ij} + b_{ij}$, $1 \leq i \leq m$, $1 \leq j \leq n$
- Množenje matrica skalarom

  Produkt λA matrice $A \in \mathbf{M}_{mn}$ i skalara $\lambda \in R$ je matrica $D \in \mathbf{M}_{mn}$ s elementima $d_{ij} = \lambda a_{ij}$, $1 \leq i \leq m$, $1 \leq j \leq n$.
- Množenje matrica

  Da bi mogao postojati umnožak matrica, one moraju biti ulančane. $A \in \mathbf{M}_{mr}$, $B \in \mathbf{M}_{sn}$ su ulančane ako je $r = s$. Produkt $A \cdot B$ dviju ulančanih matrica $A \in \mathbf{M}_{mr}$, $B \in \mathbf{M}_{rn}$ je matrica $C \in \mathbf{M}_{mn}$ s elementima

  $c_{ij} = \sum_{k=1}^{r} a_{ik} b_{kj}$, $1 \leq i \leq m$, $1 \leq j \leq n$.

Postoje razna svojstva prilikom izvođenja operacija sa matricama, no kako ona nisu usko povezana sa računanjem ranga matrice neće biti detaljnije obrađivana.

Za matricu $A \in \mathbf{M}_n$ kažemo da je regularna (invertibilna, nesingularna) ako postoji matrica $B \in \mathbf{M}_n$ takva da je $AB = BA = I$ (*). Ako matrica $C \in \mathbf{M}_n$ nije regularna, kažemo da je singularna i determinanta joj je jednaka nuli. Za regularnu matricu $A \in \mathbf{M}_n$ postoji jedinstvena matrica $B \in \mathbf{M}_n$ sa svojstvom *, takva matrica se naziva inverzna matrica matrice $A \in \mathbf{M}_n$ i označava se s $A^{-1} \in \mathbf{M}_n$. Vrijedi $AA^{-1} = A^{-1}A = I$. [13]

2.3.3. Rang matrice i elementarne transformacije matrice

Svaku matricu

$$A = [a_{ij}] = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

možemo promatrati kao niz njezinih vektora redaka

$A = [a_{ij}] = (a^1,...,a^m)$, gdje je $a^1 = (a_{11},...,a_{1n})$, ... , $a^m = (a_{m1},...,a_{mn})$.

i vektora stupaca

$A =[a_{ij}]=[a_1,...,a_n]$, gdje je $a_1 = \begin{bmatrix} a_{11} \\ \vdots \\ a_{m1} \end{bmatrix}$, . . ., $a_n = \begin{bmatrix} a_{1n} \\ \vdots \\ a_{mn} \end{bmatrix}$.

Na taj način možemo stupce i retke promatrati kao vektore te ispitivati njihovu linearnu zavisnost i uvesti vektorske prostore. Kažemo da je skup vektora $\vec{a}_1$, ..., $\vec{a}_n \in \mathbf{X_0}$ linearno zavisan onda i samo onda ako se barem jedan od njih može zapisati kao linearna kombinacija ostalih vektora.

Broj linearno nezavisnih vektora redaka matrice A naziva se rang matrice po retcima i jednak rangu matrice po stupcima, odnosno broju linearno nezavisnih stupaca matrice A. Rang matrice $A \in \mathbf{M}_{mn}$, r(A) je dakle broj linearno nezavisnih redaka (stupaca) matrice A.

Rang matrice računski određujemo koristeći se elementarnim transformacijama matrice. Pod elementarnim transformacijama matrice $A \in \mathbf{M}_{mn}$ podrazumijeva se:

- zamjena dvaju redaka (stupaca) matrice
- množenje retka (stupca) matrice skalarom $\lambda \in R$, $\lambda \neq 0$
- dodavanje nekom retku (stupcu) drugog retka (stupca)

Važno je naglasiti da elementarne transformacije nad stupcima i retcima matrice ne mijenjaju rang te matrice.

Dvije matrice $A,B \in \mathbf{M}_{mn}$ su ekvivalentne ako se jedna iz druge dobiva primjenom konačno mnogo elementarnih transformacija. Rangovi takve dvije matrice su jednaki. Rang matrice dimenzija $m \times n$ je cijeli broj između 0 i min(m,n), a jedino nul matrice imaju rang nula. Koristeći rang matrice moguće je odrediti regularnost kvadratnih matrica. Matrica $A \in \mathbf{M}_n$ je regularna onda i samo onda ako je r(A)= n. [13]

3. PROGRAMSKO RJEŠENJE

3.1. Ideja rješavanja

Prilikom izračuna ranga matrice najvažniju ulogu imaju elementarne transformacije, a sposobnost primjene istih ključ je uspješnog izračunavanja ranga. Nekada su matrice toliko složene da je potrebno puno truda i vremena kako bi se uvidjelo koje transformacije je potrebno koristiti. Jedan od načina izračunavanja ranga je i korištenje Gaussovog postupka eliminacije. Ovaj postupak je zapravo metoda rješavanja sustava linearnih jednadžbi tako da se sustav svede na njemu ekvivalentan sustav iz kojeg lako možemo očitati rješenja. Pošto se ova metoda koristi za sustave linearnih jednadžbi, a oni se mogu zapisati u matričnom obliku, sasvim je izvjesno da se postupak može koristiti i za matrice. [14] Kako postupak sadrži određena pravila koja se daju interpretirati matematički i programski, ovaj način nameće se kao najprikladniji za programsko izračuvanje ranga matrice. Primjenjujući Gaussov postupak eliminacije matrica se svodi na „*row echelon form*“ matricu i iz takve forme određuje se rang.

3.2. Postupak rješavanja

Prethodno spomenuti oblik matrice, gledajući po retcima, koji rezultira iz Gaussove metode eliminacije ima sljedeća svojstva:

- svi retci koji ne sadrže nule (ne-nul), tj. oni koji imaju barem jedan element koji nije nula, nalaze se iznad redaka koji sadrže sve nule, odnosno svi retci koji sadrže sve nule nalaze se na dnu matrice
- vodeći koeficijent, tj. prvi ne-nul element s lijeva, koji se još naziva i pivot, u ne-nul retku je uvijek strogo desno od vodećeg koeficijenta retka iznad.

Ova dva uvjeta impliciraju da su svi elementi u stupcu ispod vodećeg koeficijenta nule.

Primjer matrice u ovom obliku:

$$\begin{bmatrix} 1 & a_0 & a_1 & a_2 \\ 0 & 0 & 2 & a_3 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

Matrica se može svesti na ovakav oblik i po stupcima, no za potrebe ovog programa bit će korišten oblik po retcima. [15]

Program funkcionira na način da se zadanoj matrici provjeravaju elementi glavne dijagonale i to po sljedećim pravilima:

- Ako matrica sadrži više redaka nego stupaca, zbog jednostavnijeg računanja matrica se transponira.
- Ukoliko se na glavnoj dijagonali nalazi ne-nul element sve elemente u tom stupcu treba učiniti nulom tako što se pronađe odgovarajući množitelj i pomnoži sa svim retcima te matrice. To se ponavlja za cijelu dijagonalu.
- Ukoliko se na dijagonali već nalazi element nula onda postoje dvije mogućnosti:
    1) ako ispod ima redak sa ne nul elementom u istom tom stupcu, ta dva retka mijenjaju mjesta
    2) ako su u tom trenutnom stupcu već svi elementi nula, taj stupac se mijenja sa zadnjim stupcem u matrici kako ne bi utjecali na ostale elemente na dijagonali, pritom pazeći na broj stupaca u matrici.

Kada se prođu svi elementi na glavnoj dijagonali, iscrpe sve mogućnosti i dobiju nule na potrebnim mjestima, prebrojavanjem ne-nul elemenata na glavnoj dijagonali matrice dobivamo rang te matrice.

3.3. Opis programskog rješenja

Program se sastoji od šest funkcija, pet računskih i glavne, main() funckije. U ovom poglavlju bit će opisane sve funkcije, redom kojim se pojavljuju unutar funkcije za izračunavanje ranga `float rangMatrice(float mat[N][N],int red,int stup).` Ova funkcija prima matricu[N][N] gdje N označava maksimalan broj redaka, odnosno stupaca i definiran je predprocesorskom naredbom #define N 100. Pošto je program namijenjen računanju ranga matrica nekih realnih veličina odabran je maksimalan broj 100, no taj broj se vrlo lako može promijeniti. Nadalje, funkcija prima stvaran broj redaka i stupaca zadane matrice.

Prvi uvjet koji se provjerava je ima li matrica više redaka nego stupaca i ukoliko je tako matrica se transponira, odnosno mijenjaju joj se retci i stupci.

```
if (red>stup)
{
    transponiranje(mat, red, stup);
    int p = red;
    red = stup;
    stup = p;
}
int pozicija = stup - 1, i, j, k;
```

Sl. 3.1. Provjera uvjeta za transponiranje matrice

Unutar ovog koda program dolazi do prve funkcije, a to je funkcija transponiranje.

```
void transponiranje(float mat[N][N], int red, int stup)
{
    int p[N][N], i, j;
    for (i = 0; i<red; i++)
        for (j = 0; j<stup; j++)
            p[j][i] = mat[i][j];

    for (i = 0; i<stup; i++)
        for (j = 0; j<red; j++)
            mat[i][j] = p[i][j];

    printf("\nTransponirana matrica:");
    printf("\n");
    ispis(mat, stup, red);
}
```

Sl. 3.2. Funkcija za transponiranje matrice

Funkcija prima iste parametre kao i prethodna funkcija. Transponiranje se vrši na način da se zada privremena matrica p[N][N], prolazi se po retcima i stupcima matrice, u privremenu matricu se spremaju određeni elementi te se potom iz privremene matrice spremaju nazad u zadanu matricu na odgovarajuće pozicije. Nakon toga se ispisuje zadana matrica pomoću funkcije `ispis`.

Funkcija `ispis` iduća je korištena funkcija. Predaju joj se redom matrica, broj redaka i broj stupaca i pomoću naredbe `printf` ispisuju. Vidljivo je da su unutar funkcije transponiranje retci i stupci predani u obrnutom redoslijedu jer im se broj nakon transponiranja promijenio. Pozicija je varijabla koja je bitna prilikom zamjene stupaca, a označava indeks zadnjeg stupca u matrici.

Nakon što se obavi transponiranje potrebno je pomoću privremene varijable zamijeniti broj redaka i stupaca zbog nesmetanog daljnjeg korištenja. Kada je matrica transponirana ulazi se u for petlju i provjeravaju se uvjeti iz algoritma. Prvi od njih prikazan je na slici 3.3.

```
if (mat[i][i])
{
    for (j = 0; j<red; j++)
    {
        if (j != i)
        {
            if (mat[j][i])
            {
                double mnozitelj = (double)mat[j][i] / mat[i][i];
                for (k = 0; k<stup; k++)
                    mat[j][k] -= mnozitelj * mat[i][k];
            }
        }
    }
}
```

Sl. 3.3. Uvjet za provjeru elemenata na dijagonali

Ako element na glavnoj dijagonali nije nula program ulazi u još jednu petlju, naredbom `j!=i` osigurava da se ne dira redak elementa kojeg trenutno pratimo te ako traženi element u stupcu `mat[j][i]` nije nula traži množitelja. Množitelj se dobiva tako što se element nad kojim obavljamo radnju, tj. element u stupcu ispod elementa na dijagonali kojeg želimo pretvoriti u nulu, podijeli sa elementom na dijagonali. Nakon toga se petljom svim elementima redaka ispod ili iznad odabranog elementa na dijagonali od njihove vrijednosti oduzima vrijednost množitelja pomnoženog sa vrijednošću elementa u njihovom stupcu i u retku u kojem se nalazi element glavne dijagonale kojeg pratimo. Na taj način dobivamo nule u stupcu u kojem se nalazi element kojeg pratimo.

Na primjeru bi to izgledalo ovako:

Zadana matrica:

$$\begin{bmatrix} 2 & -3 & 16 & 1 \\ 1 & 6 & -2 & 3 \\ 1 & 3 & 2 & 2 \end{bmatrix}$$

Element na glavnoj dijagonali je 2 što je različito od nule. Nalazimo se u prvom retku, pa je indeks retka `i` = 0, kako ne bi dirali prvi redak i zbog uvjeta `j!=i`, `j` postavljamo na 1 i ulazimo u drugi

redak. Prvi element tog retka nije 0, pa se računa množitelj. U ovom slučaju za drugi redak množitelj je $\frac{1}{2}$ = 0.5. Ulazi se u for petlju i svakom elementu drugog retka oduzima se umnožak množitelja i elementa iznad njega. Znači redom slijedi: 1 – 0.5 · 2= 0, 6 – 0.5 · (-3) = 7.5, -2 – 0.5 · 16 = -10, 3 – 0.5 · 1 = 2.5. Izlazimo iz petlje, `j` se povećava za 1 i ulazimo u idući redak, gdje dobivamo isti množitelj i rješenja 0, 4.5, -6 i 1.5, pa matrica nakon toga izgleda ovako:

$$\begin{bmatrix} 2 & -3 & 16 & 1 \\ 0 & 7.5 & -10 & 2.5 \\ 0 & 4.5 & -6 & 1.5 \end{bmatrix}$$

Isti postupak ponavlja se za idući redak, odnosno za idući element na dijagonali i dobije se ovakvo rješenje:

$$\begin{bmatrix} 2 & 0 & 2 & 12 \\ 0 & 7.5 & -10 & 2.5 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Kako je idući element na dijagonali nula više se ne provodi ovaj dio koda nego se prelazi na idući slučaj. Iako je već sada vidljivo koji je rang matrice, program prelazi na sljedeći dio i provodi dio koda za slučaj kad je na dijagonali nula. Kod je prikazan na slici 3.4.

```
else
{
    bool zamjena = false;
    for (j = i + 1; j<red; j++)
    {
        if (mat[j][i])
        {
            zamjenaRetka(mat, i, j, stup);
            zamjena = true;
            break;
        }
    }
    if (!zamjena)
    {
        if (pozicija>i)
        {
            zamjenaStupca(mat, i, pozicija, red);
            pozicija--;
        }
        else
            break;

    }
    i--;
}
```

Sl. 3.4. Dio koda za slučaj da je na glavnoj dijagonali nul element

Prvo se postavlja varijabla zamjena na false i pomoću te varijable prati se je li neki redak/stupac već mijenjan kako se ne bi ponavljale iste radnje. Ulazi se u petlju, `j` se automatski povećava za jedan da se prvi redak ne provjerava jer ukoliko se nula nalazi na dijagonali u prvom retku nema smisla i potrebe da se redak uspoređuje i mijenja sam sa sobom, a ukoliko se radi o nekom drugom retku, prvi redak i prvi stupac su već sređeni sa prethodnim dijelom koda, pa se oni više ne diraju. Nakon što se uđe u petlju provjeri se jesu li elementi u retcima ispod dijagonale nule i ukoliko nisu mijenja se prvi redak u kojem element nije nula sa retkom u kojem je element na dijagonali bio nula. Zamjenu postavljamo na true i izlazi se iz petlje te se nastavlja na idući element na dijagonali. U opisanom slučaju nailazimo na funkciju `float zamjenaRetka.`

```
void zamjenaRetka(float mat[N][N], int red1, int red2, int stup)
{
    int i, p;
    for (i = 0; i<stup; i++)
    {
        p = mat[red1][i];
        mat[red1][i] = mat[red2][i];
        mat[red2][i] = p;
    }
}
```

Sl. 3.5. Funkcija za zamjenu redaka

Ovoj funkciji predaje se matrica, indeks retka kojeg želimo zamijeniti, `red1`, indeks retka s kojim ga mijenjamo, `red2` i broj stupaca da se zna do kud ide petlja. U petlji se u privremenu varijablu `p` spremaju elementi jednog retka, zatim se elementima istog tog retka dodaju elementi retka s kojim ga želimo zamijeniti i na poslijetku se elementi iz privremene varijable pridruže retku s kojim mijenjamo prvi red.

Ako se prilikom provjere za mijenjanje redaka u retcima ispod elementa glavne dijagonale koji je nula nalaze također nule, nije moguće obaviti zamjenu po retcima. Varijabla `zamjena` ostaje false i ispunjava se uvjet `if(!zamjena)`. Ukoliko je pozicija takva da se element na dijagonali nalazi u zadnjem retku i zadnjem stupcu, tj. ako se radi o kvadratnoj matrici, nije moguće mijenjati retke jer su nad svim retcima već obavljene potrebne radnje i nema preostalih redaka, prekida se izvođenje programa. To se ostvaruje tako što se varijabla `pozicija` postavlja na vrijednost indeksa zadnjeg stupca u matrici i uspoređuje se sa varijablom `i` koja označava indeks retka. Ukoliko je indeks retka veći od `i` obavlja se zamjena stupaca pomoću funkcije `float zamjenaStupaca.` Funkcija je gotovo identična onoj za zamjenu redaka, pa ju nije potrebno objašnjavati. Kada se izmjene stupci varijabla `pozicija` se smanjuje jer nakon promjene stupac koji je zamijenjen više nije dostupan za zamjenu. Vrlo je važno nakon što se zamjene retci varijablu `i` smanjiti za jedan kako bi se nakon zamjene redaka ili stupaca program ponovno vratio do elementa na dijagonali koji je prije zamjene bio nula te obavio operacije nad ostalim elementima u tom stupcu koji nisu bili zahvaćeni promjenom retka/stupca.

Unutar glavne petlje nalazi se još `ispis` koji nakon svakog koraka ispisuje rezultat, pa je tako moguće vidjeti sve korake izračuna ranga. Nakon što su obavljene sve operacije nad matricom, potrebno je odrediti koliki je rang.

```
int rang = 0;
for (i = 0; i<red; i++)
{
    if (mat[i][i])
        rang++;
    else
        break;
}

printf("\n");
return rang;
```

Sl. 3.6. Određivanje ranga matrice

Rang se računa tako da se prolazi kroz matricu i gleda elemente glavne dijagonale. Ako su elementi glavne dijagonale različiti od nule povećava se varijabla `rang`. Ta varijabla je brojač prvotno postavljen na nulu, a ujedno i vrijednost ranga. Kada dođe do elementa na dijagonali koji je nula petlja se prekida zbog pravila da se svi retci sa nulama nalaze ispod redaka koji sadrže barem jedan ne-nul element. Funkcija `rangMatrice` na poslijetku vraća cjelobrojnu varijablu `rang`.

Prilikom pokretanja programa postoje dvije mogućnosti za korisnike. Nakon što unesu dimenzije matrice, korisnici odabiru žele li sami unijeti elemente ili žele izračunati rang nasumično generirane matrice. Nasumično generiranje omogućeno je prvenstveno radi brže provjere točnosti programa prilikom testiranja u Symbolabu. [16]

```
do {
    printf("\nOdaberite:\n 1 - samostalni unos elemenata\n"
           "2 - nasumicno generirani elementi\nOdabir: ");
    scanf("%d", &a);
} while ((a != 1) && (a != 2));
switch (a) {
case 1:
    printf("\nUnesite elemente matrice: \n");
    for (i = 0; i<red; i++)
        for (j = 0; j<stup; j++)
            scanf("%f", &mat[i][j]);
    break;
case 2:
    srand(time(NULL));
    for (i = 0; i<red; i++)
        for (j = 0; j<stup; j++)
            mat[i][j] = rand() % 30 - 10;
    break;
}
```

Sl. 3.7. Primjer koda za unos ili generiranje elemenata matrice

3.4. Primjer korištenja programa

```
Unesite:
 broj redaka: 3
 broj stupaca: 4

Odaberite:
 1 - samostalni unos elemenata
 2 - nasumicno generirani elementi
Odabir: 1

Unesite elemente matrice:
2 -3 16 1 1 6 -2 3 1 3 2 2

Matricni zapis:

2.00 -3.00 16.00 1.00
1.00 6.00 -2.00 3.00
1.00 3.00 2.00 2.00

Koraci rjesenja:

2.00 -3.00 16.00 1.00
0.00 7.50 -10.00 2.50
0.00 4.50 -6.00 1.50

2.00 0.00 12.00 2.00
0.00 7.50 -10.00 2.50
0.00 0.00 0.00 0.00

2.00 0.00 2.00 12.00
0.00 7.50 2.50 -10.00
0.00 0.00 0.00 0.00


Rang zadane matrice je: 2

--------------------------------
Process exited after 13.26 seconds with return value 0
Press any key to continue . . .
```

Sl. 3.8. Primjer korištenja programa prilikom samostalnog unosa elemenata matrice

```
Unesite:
 broj redaka: 6
 broj stupaca: 4

Odaberite:
 1 - samostalni unos elemenata
 2 - nasumicno generirani elementi
Odabir: 2

Matricni zapis:

5.00 12.00 17.00 -1.00
19.00 15.00 9.00 14.00
-2.00 6.00 16.00 -9.00
-8.00 -8.00 4.00 17.00
1.00 12.00 2.00 13.00
14.00 2.00 2.00 -5.00

Transponirana matrica:
5.00 19.00 -2.00 -8.00 1.00 14.00
12.00 15.00 6.00 -8.00 12.00 2.00
17.00 9.00 16.00 4.00 2.00 2.00
-1.00 14.00 -9.00 17.00 13.00 -5.00

Koraci rjesenja:

5.00 19.00 -2.00 -8.00 1.00 14.00
0.00 -30.60 10.80 11.20 9.60 -31.60
0.00 -55.60 22.80 31.20 -1.40 -45.60
0.00 17.80 -9.40 15.40 13.20 -2.20

5.00 0.00 4.71 -1.05 6.96 -5.62
0.00 -30.60 10.80 11.20 9.60 -31.60
0.00 0.00 3.18 10.85 -18.84 11.82
0.00 0.00 -3.12 21.92 18.78 -20.58

5.00 0.00 0.00 -17.12 34.88 -23.13
0.00 -30.60 0.00 -25.69 73.67 -71.78
0.00 0.00 3.18 10.85 -18.84 11.82
0.00 0.00 0.00 32.56 0.29 -8.98

5.00 0.00 0.00 0.00 35.03 -27.85
0.00 -30.60 0.00 0.00 73.90 -78.86
0.00 0.00 3.18 0.00 -18.94 14.81
0.00 0.00 0.00 32.56 0.29 -8.98


Rang zadane matrice je: 4
```

Sl. 3.9. Primjer korištenja programa za nasumične elemente

4. ZAKLJUČAK

U ovom diplomskom radu ostvaren je cilj zadan u zadatku diplomskog rada. Program uspješno računa rang zadane mu matrice. Na samom početku detaljno je objašnjen programski jezik C kako bi čitatelji imali uvid u koncept programskog jezika, ali i u samo programiranje. Prvo je objašnjena povijest nastanka jezika, a zatim njegove karakteristike i nedostaci. Mnoge od tih karakteristika korištene su prilikom izrade programa i za njegovo razumijevanje bitno je razumjeti kako funkcioniraju. Zatim se čitatelja upoznaje sa linearnom algebrom. Linearna algebra je nešto složenija matematička disciplina i zbog svoje apstraktne prirode ljudima je često teško shvatljiva. No postoje dijelovi ove discipline koji su ipak bliži ljudima. Takve su matrice koje su, za razliku od npr. vektorskih prostora, lakše shvatljive strukture budući da ih se može jednostavno zapisivati i svaka operacija nad njima ima odraz na samu matricu te je vidljiva čitatelju. Kako je kod programskog jezika vrlo bitno razumijevanje karakteristika jezika, tako je za matrice vrlo bitno poznavanje i korištenje elementarnih transformacija koje su opisane u posljednjem teorijskom poglavlju i upravo se od njih kreće prilikom kreiranja ideje za pisanje programskog rješenja.

Glavni cilj zadatka ostvaren je pisanjem algoritma, odnosno iznošenjem ideje za rješavanje problema u istoimenom poglavlju. Algoritam se temelji na matematičkom postupku Gaussove eliminacije i svođenjem matrice na formu iz koje se lako očitava rang matrice. Dolazak do te ideje zapravo je, uz programsku implementaciju iste, bio najveći izazov prilikom rješavanja problema. Jednom kada su postavljena pravila iz njih je napisan algoritam te kasnije program koji po koracima prati algoritam i rješava problem. Program je zatim testiran na brojnim primjerima. Korištenjem Symbolab-a, postojećeg online alata za izračun ranga, rezultati su provjeravani i nije pronađena nikakva greška iz čega zaključujem da je problem uspješno riješen. Moguća su poboljšanja u vidu dodavanja grafičkog sučelja ili izrade mobilne aplikacije za računanje ranga matrice, no algoritam ostaje isti.

LITERATURA

[1] Wikipedia, C (programski jezik), https://hr.wikipedia.org/wiki/C_(programski_jezik) (stranica posjećena: 24. lipnja 2018.)

[2] Dennis M. Ritchie, Brian W. Kernighan: Programski jezik C, CET, 2003.

[3] Tutorialspoint, simple easy learning, C tutorial, https://www.tutorialspoint.com/cprogramming (stranica posjećena: 24. lipnja 2018.)

[4] Sveučilište u Rijeci, Tehnički fakultet, Programiranje, Uvod u pokazivače, http://www.riteh.uniri.hr/zav_katd_sluz/zr/nastava/programiranje/download/predavanja/uvod_u_pokazivace.pdf (stranica posjećena: 25. lipnja 2018.)

[5] Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet, Predavanje pointeri i strukture https://web.math.pmf.unizg.hr/nastava/mreze/slideovi/vjezbe_02_pointeri_strukture.pdf (stranica posjećena: 25. lipnja 2018.)

[6] Wikipedia, stdio.h, https://hr.wikipedia.org/wiki/Stdio.h (stranica posjećena: 25. lipnja 2018.)

[7] Wikipedia, Kompajler, https://bs.wikipedia.org/wiki/Kompajler (stranica posjećena: 25. lipnja 2018.)

[8] Wikipedia, Debugger, https://sh.wikipedia.org/wiki/Debugger (stranica posjećena: 25. lipnja 2018.)

[9] Wikipedia, Microsoft Visual Studio, https://en.wikipedia.org/wiki/Microsoft_Visual_Studio (stranica posjećena: 25. lipnja 2018.)

[10] Wikipedia, Linearna algebra, https://hr.wikipedia.org/wiki/Linearna_algebra (stranica posjećena: 26. lipnja 2018.)

[11] Damir Bakić: Linearna algebra, Školska knjiga Zagreb, 2008.

[12] CARNet Loomen, FERIT, kolegij Linearna algebra, predavanje pojam matrice https://loomen.carnet.hr/ (stranica posjećena: 26. lipnja 2018.)

[13] Sveučilište u Osijeku, Odjel za matematiku, kolegij Linearna algebra 1, skripta u PDF formatu, https://www.mathos.unios.hr/images/homepages/scitowsk/LA1.pdf (stranica posjećena: 26. lipnja 2018.)

[14] Sveučilište u Splitu, Fakultet elektrotehnike, strojarstva i brodogradnje, Matematika predavanja, http://lavica.fesb.hr/mat1/predavanja/node32.html (stranica posjećena: 26. kolovoza 2018.)

[15] Wikipedia, Row echleon form, https://en.wikipedia.org/wiki/Row_echelon_form (stranica posjećena: 26. kolovoza 2018.)

[16] Symbolab, https://www.symbolab.com/ (stranica posjećena: 31. kolovoza 2018.)

SAŽETAK

Cilj ovoga rada bio je izraditi program za računanje ranga matrice u programskom jeziku C u svrhu olakšavanja računanja istoga. Teorijski dio rada obuhvaća detaljan opis programskog jezika C, njegovu povijest i sve funkcionalnosti koje je nužno poznavati za rad u programskom jeziku. Opisano je i razvojno okruženje Microsoft Visual Studio u kojemu je program napisan. Nadalje, rad sadrži opis matematičke grane linearne algebre koja se, između ostalog, bavi matricama. Opisano je što je to algebra i čemu služi te je detaljnije objašnjena algebra matrica čije je razumijevanje ključno za uspješno obavljanje zadatka. U zadnjem dijelu rada objašnjen je sam program, sve njegove funkcije i naredbe pomoću kojih se dolazi do konačnog rezultat, odnosno do ranga zadane matrice. Korisnik prilikom pokretanja programa unosi dimenzije i elemente matrice, a program mu vraća brojčanu vrijednost ranga matrice.

Ključne riječi: C, linearna algebra, matrice, Microsoft Visual Studio, rang matrice

ABSTRACT

The goal of this project was to develop a program for calculating the rank of a matrix in programming language C. The theoretical part of this assignment includes a detailed description of the C programming language, its history and all its functionalities that are necessary to know in order to work with the language and description of Microsoft Visual Studio which was used to develop the program. Furthermore, it contains description of linear algebra, part of mathematics which, among other things, deals with matrices. It is described what algebra is and what it is used for. The matrix algebra is explained in more detail because its understanding is crucial to successfully perform the task. The program, all its functions and commands used to reach the final result, the rank of matrix, are explained in the last part of the assignment. When the user initiates the program, it inputs dimensions and elements of a matrix and then the program returns the numeric value of the rank of a matrix.

Keywords: C, linear algebra, matrices, Microsoft Visual Studio, the rank of a matrix

ŽIVOTOPIS

Iva Pandurić rođena je 20. kolovoza 1994. godine u Osijeku, Hrvatska. Godine 2001. započinje osnovnoškolsko obrazovanje u OŠ Matija Gubec Magadenovac, gdje se iskazala na brojnim županijskim natjecanjima, sa značajnijim rezultatima iz predmeta matematika i geografija. Nakon završene osnovne škole, 2009. godine upisuje opću gimnaziju u Valpovu. U gimnaziji nastavlja nizati uspjehe na natjecanjima iz ranije spomenutih predmeta, posebice iz geografije i povijesti gdje je, kao i u ranijem obrazovanju, više godina za redom postizala rezultate među najboljih 5 učenika u županiji. Sa odličnim uspjehom tijekom cijelog školovanja i položenom državnom maturom završava dotadašnje obrazovanje. Preddiplomski sveučilišni studij računarstva na Elektrotehničkom fakultetu u Osijeku upisuje 2013. godine. Isti završava 2016. godine sa najvišom pohvalom te upisuje diplomski studij na istom fakultetu, smjer programsko inženjerstvo. Tijekom fakultetskog obrazovanja prima stipendiju Nacionalne zaklade za potporu učeničkom i studentskom standardu kao nadareni student koji se obrazuje za deficitarno zanimanje. Od ostalih znanja i vještina posjeduje znanje engleskog jezika, vrlo dobro poznaje rad na računalu te ima vozačku dozvolu B kategorije.

Potpis:

Iva Pandurić