Kakuro u programskom jeziku C

Maričević, Ivan

Undergraduate thesis / Završni rad

2016

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

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:183868

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

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

Stručni studij

KAKURO U PROGRAMSKOM JEZIKU C

Završni rad

Ivan Maričević

Osijek, 2016.

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

PROTOKOL OBRANE ZAVRŠNOG RADA NA PREDDIPLOMSKOM STRUČNOM STUDIJU

"Otvaram postupak obrane završnog rada pristupnika: Ivan Maričević

koji će pred Povjerenstvom u sastavu:

Predsjednik: Izv.prof.dr.sc. Dominika Crnjac-Milić

Mentor: Doc.dr.sc. Tomislav Rudec

Član: Doc.dr.sc. Alfonzo Baumgartner

---

braniti svoj rad pod naslovom: Kakuro u programskom jeziku C

Potom predsjednik čita životopis koji je u prilogu.

"Molim pristupnika da u trajanju 10-15 minuta iznese u kratkim crtama sažetak svog rada. Izvolite!"

Predsjednik, mentor, član Povjerenstva i auditorij sjedaju.
Kada je pristupnik završio izlaganje, predsjednik se obraća mentoru:
" Doc.dr.sc. Tomislav Rudec , *molim Vas da Vi kao mentor prvi postavite pitanja."*

Zatim predsjednik poziva člana Povjerenstva da postavi pitanja, te na kraju i sâm postavlja pitanja pristupniku.
Nakon što je pristupnik odgovorio na sva pitanja, predsjednik se obraća auditoriju:

"Želi li netko od nazočnih postaviti pitanje vezano za ovaj rad?"
"Ako nitko više nema što pitati ili dodati, pozivam Povjerenstvo na vijećanje kako bi za desetak minuta donijeli odluku."

Dvoranu napuštaju pristupnik i auditorij, ili Povjerenstvo.
Nakon donošenja odluke Povjerenstvo poziva pristupnika.

"Pristupnik Ivan Maričević obranio je završni rad na temu: Kakuro u programskom jeziku C
i polučio ocjenu pismenog dijela završnog rada: Izvrstan (5) ,
ocjenu usmenog dijela završnog rada: Vrlo dobar (4),
Time je ukupna ocjena završnog rada: Izvrstan (5) ,
te je pristupnik Ivan Maričević stekao stručno zvanje Stručni prvostupnik (baccalaureus) inženjer elektrotehnike

smjer Informatika

Molim pristupnika da potpiše zapisnik i da mu čestitamo na uspjehu sa željom da nastavi vrijedno raditi u praksi i da ime Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek dostojno promiče svojim znanjem i zalaganjem."

Pristupnik i članovi Povjerenstva potpisuju zapisnik.

"Ovime je postupak obrane završen."

Povjerenstvo, pristupnik i auditorij napuštaju dvoranu

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 04.10.2016.

Ime i prezime studenta:	Ivan Maričević
Studij:	Preddiplomski stručni studij Elektrotehnika, smjer Informatika
Mat. br. studenta, godina upisa:	AI 4177, 25.07.2012.
Ephorus podudaranje [%]:	6 %

Ovom izjavom izjavljujem da je rad pod nazivom**: Kakuro 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:

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

IZJAVA

Ja, Ivan Maričević, OIB: 10072501948, student/ica na studiju: Preddiplomski stručni studij Elektrotehnika, smjer Informatika, dajem suglasnost Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek da pohrani i javno objavi moj **završni rad**:

Kakuro u programskom jeziku C

u javno dostupnom fakultetskom, sveučilišnom i nacionalnom repozitoriju.

Osijek, 04.10.2016.

_____
potpis

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1S: Obrazac za imenovanje Povjerenstva za obranu završnog rada na preddiplomskom stručnom studiju

Osijek, 22.09.2016.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu završnog rada na preddiplomskom stručnom studiju

Ime i prezime studenta:	Ivan Maričević
Studij, smjer:	Preddiplomski stručni studij Elektrotehnika, smjer Informatika
Mat. br. studenta, godina upisa:	AI 4177, 25.07.2012.
OIB studenta:	10072501948
Mentor:	Doc.dr.sc. Tomislav Rudec
Sumentor:	Doc.dr.sc. Alfonzo Baumgartner
Predsjednik Povjerenstva:	Izv.prof.dr.sc. Dominika Crnjac-Milić
Član Povjerenstva:	Doc.dr.sc. Alfonzo Baumgartner
Naslov završnog rada:	Kakuro u programskom jeziku C
Znanstvena grana rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak završnog rada	Tema je zauzeta
Prijedlog ocjene pismenog dijela ispita (završnog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 Postignuti rezultati u odnosu na složenost zadatka: 3 Jasnoća pismenog izražavanja: 2 Razina samostalnosti: 3
Datum prijedloga ocjene mentora:	22.09.2016.

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

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

Obrazac Z2S: Zapisnik s obrane završnog rada na stručnom studiju

Osijek, 29.9.2016.

Zapisnik s obrane završnog rada na stručnom studiju

Ime i prezime studenta:	Ivan Maričević
Studij, smjer:	Preddiplomski stručni studij Elektrotehnika, smjer Informatika
Mat. br. studenta, godina upisa:	AI 4177, 25.07.2012.
Mentor:	Doc.dr.sc. Tomislav Rudec
Sumentor:	Doc.dr.sc. Alfonzo Baumgartner
Predsjednik Povjerenstva:	Izv.prof.dr.sc. Dominika Crnjac-Milić
Član Povjerenstva:	Doc.dr.sc. Alfonzo Baumgartner
Naslov diplomskog rada:	Kakuro u programskom jeziku C
Kratko obrazloženje prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 Postignuti rezultati u odnosu na složenost zadatka: 3 Jasnoća pismenog izražavanja: 2 Razina samostalnosti: 3

Obrana završnog rada održana je na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek dana 29.9.2016. u 10.00 sati sati

Pitanja članova Povjerenstva za obranu:	Rudec 1: Usporedi sudoku i kakuro 2. koji programski jezik koristiti za čisto programiranje? Baumgartner: 1. Koja se funkcija prva izvodi? 2. Razlika matrice i polja? Crnjac: 1. Može li se dogoditi beskonačno rješavanje? 2. Gdje ste uzeli primjere za testiranje programa?
Mišljenje mentora o pismenom dijelu rada i Povjerenstva o tijeku obrane:	Student je sve napravio sam i riješio zadatak bolje nego što se očekivalo. Student je samostalno i točno izradio rad, prezentirao ga i odgovorio na postavljena mu pitanja.

Ocjena pismenog dijela ispita (završni rad):	Izvrstan (5)
Ocjena usmenog dijela ispita (obrana završnog rada):	Vrlo dobar (4)
Ukupna ocjena na diplomskom ispitu:	**Izvrstan (5)**

Predsjednik Povjerenstva:	
Mentor:	
Član:	
Zapisničar:	
Pristupnik:	

Uspješnom obranom završnog rada pristupnik stječe stručni naziv:

Stručni prvostupnik (baccalaureus) inženjer elektrotehnike smjer Informatika

Sadržaj

1. Uvod **1**

1.1 Zadatak završnog rada 1

2. Kakuro i program **2**

2.1 Kakuro 2

2.1.1 Definicija i povijest 2

2.1.2 Opis igre i pravila 2

2.1.3 Tehnike rješavanja 3

2.1.4 Kombinacije 5

2.2 Program 5

3. Algoritmi i njihova implementacija **8**

3.1 Zadnja nula 8

3.2 Križanje 9

3.4 Višestruko križanje 12

3.5 Eliminacija 15

4. Pomoćne funkcije **19**

4.1 Kombinacije 19

4.1.1 Funkcija Kombinacije 19

4.1.2 Funkcija Broj 19

4.1.3 Funkcija Dodaj red 21

4.2 3D polja 21

4.3 Ostale funkcije 24

5. Zaključak 26

1. UVOD

Logičke igre su igre koje imaju određena pravila i mogu se riješiti logikom. Kakuro je jedna od tih igara i posebna je po tome što uz logiku zahtjeva i kombinatoriku. Zadatak rada je proučavanje Kakura i tehnika rješavanja te njihova implementacija uz pomoć programskog jezika C. Potrebna je i izrada programa koji s tim tehnikama sam može riješiti zadani Kakuro. Rad se sastoji od tri cjeline: proučavanje Kakura i algoritama, razrada algoritama i programiranje te pisanje rada. Prva dva dijela su u potpunosti opisana u ovom pisanom dijelu. Pisani dio se sastoji od tri poglavlja: Kakuro i program, algoritmi i njihova implementacija i pomoćne funkcije.

U poglavlju „Kakuro i program“ nalazi se opis same igre, njezina pravila i tehnike rješavanja kao i osnovni opis programa, tj. njegov način rada i pojednostavljen grafički prikaz.

Poglavlje „Algoritmi i njihova implementacija“ sadrže obradu tehnika rješavanja i njihov prikaz uz pomoć programskog jezika C.

Zadnje poglavlje „Pomoćne funkcije“, razradit će pomoćne funkcije potrebne za rad programa.

1.1 Zadatak završnog rada

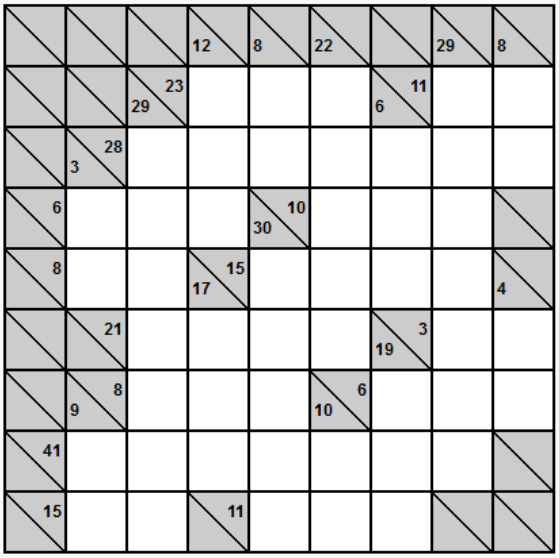
Proučavanje algoritama korištenih za rješavanje igre logičke Kakuro i njihova implementacija pomoću programskog jezika C.

2. KAKURO I PROGRAM

2.1 Kakuro

2.1.1 Definicija i povijest Kakura

[1]Kakuro, Cross Sums ili Kakro je vrsta logičke igre koja se često opisuje kao matematička križaljka. Prvi kakuro se zvao Cross Sums i objavljen je 1966. godine od Dell Magazines. Nakon toga se redovito pojavljivao u matematičkim i logičkim objavama. U Japan je stigao tek 1980. godine gdje ga je doveo Maki Kaji, predsjednik Nikoli igara. Prvo mu je dao ime Kasan Kurosu, što je kombinacija japanskih riječi za zbrajanje i križanje. Šest godina kasnije Nikoli mu je promijenio ime u Kakro i izdao prvu Kakro/Kakuro knjižicu. Nakon toga Kakuro je postao izuzetno popularna logička igra u Japanu do 1992. godine kada ju je zamijenio Sudoku. Kakuro se nastavio širiti i trenutno ga se može pronaći u skoro svim časopisima i dnevnim novinama.



S1. 2.1. Kakuro srednje težine dimenzije 8x8

2.1.2 Opis i pravila igre

Kakuro se sastoji od mreže polja od kojih neka služe za upis dok su druga već ispunjena. Polja koja služe za upis se nazivaju „bijela“ polja i ona služe za upis jednog broja, a polja koja su prethodno ispunjena nazivaju „crna“ polja. Ona mogu sadržavati ništa, jedan broj i dva broja. Ukoliko „crno“ polje sadrži jedan ili dva broja podijeljeno je na dva dijela dijagonalom.

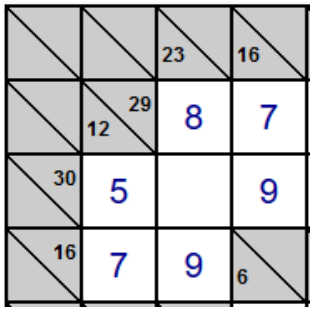
[1] http://www.kakurolive.com/about-kakuro.php

Broj u „crnom“ polju označava zbroj brojeva koji se moraju nalaziti u „bijelim“ poljima ispod ili desno od njega. Smjer popunjavanja ovisi o položaju broja tako što broj ispod dijagonale usmjerava prema dolje ,a broj iznad dijagonale prema desno. Ti brojevi se često zovu „tragovi“. Cilj igre je upisati brojeve od 1 do 9 u „bijela“ polja mreže na takav način da se ne ponavljaju u tom stupcu i retku.

2.1.3 Tehnike rješavanja

Prilikom rješavanja Kakura koristi se kombinacija kombinatorike i logike te postoji više različitih tehnika.

Zadnje polje

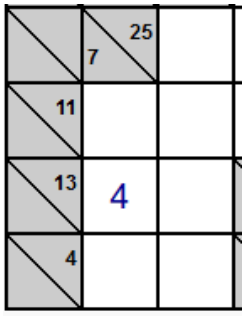


S1. 2.2. Primjer zadnjeg kvadrata

Zadnje prazno polje u retku ili stupcu može se lako riješiti oduzimanjem. Da bi se dobilo rješenje vrši se oduzimanje brojeva ispunjenih polja iz stupca ili retka od broja „crnog“ polja. U ovom slučaju to bi bilo 23-(8+9)=6.

Križanje

Tehnika koja se zasniva na traženju jedinog mogućeg broja pomoću kombinatorike. Na križanju između dva „traga“ uspoređuju se brojevi iz kombinacija mogućih za to mjesto i ukoliko postoji samo jedan zajednički broj on se i upisuje.

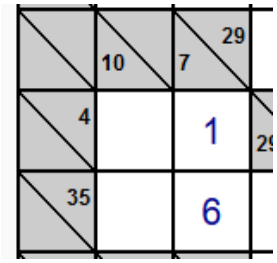


S1. 2.3. Primjer križanja

Na slici 2.3. postoje tri kombinacije brojeva za „trag“ 13, to su 4+9, 5+8, i 6+7, a za „trag“ 7 je samo jedna i to 1+2+4. Prema tome „trag“ 13 pokazuje da je u polje moguće je upisati brojeve 4, 5, 6, 7, 8 i 9, a „trag“ 7 da je moguće upisati brojeve 1, 2 i 4. Kada se usporede ta dva niza vidi se da je na to mjesto moguće upisati samo broj 4 koji se onda i upisuje.

Višestruko križanje

Radi na principu provjere mogućih kombinacija za svaki kvadrat nekog „traga“ i izbacivanju kombinacija dok ne ostane samo jedna.

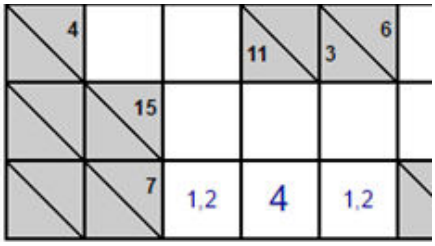


S1. 2.4. Primjer korištenja višestrukog križanja

U slučaju na slici 2.4 vrši se provjera „traga“ 7. Moguće kombinacije za njega su 1+6, 2+5 i 3+4. Prvi red koji sječe taj stupac je red „traga“ 4 za kojeg su moguće kombinacije 1+3 što znači da se može izbaciti kombinacija 2+5 pošto se ni jedan broj ne pojavljuje. Drugi red koji sječe je red „traga“ 35, a ima samo jednu kombinaciju i to je 5+6+7+8+9 te na osnovu toga možemo izbaciti kombinaciju 3+4. Sada nam preostaje samo kombinacija 1+6 i nakon toga uz pomoć tehnike križanja lako možemo odrediti koji broj se nalazi u kojem polju.

Eliminacija

Tehnika eliminacije se primjenjuje ukoliko se u jednom redu u dva polja pojavljuje mogućnost dva ista broja. Može se zamisliti da su ta dva polja ispunjena s tim brojevima te potom izvršiti jedna od tehnika. Isto vrijedi i za veći broj slobodnih polja.



S1. 2.5. Primjer korištenja tehnike eliminacije

U primjeru, u redu „traga“ 7, na prvo i zadnje polje moguće je staviti samo brojeve 1 i 2. Iako ne znamo u koje polje mora ići koji broj znamo da se ti brojevi ne mogu nalaziti u ostalim poljima, tj. da se brojevi 1 i 2 ne mogu nalaziti u srednjem polju. Potom zaključujemo da se u srednjem polju može nalaziti samo broj 4.

2.1.4. Kombinacije

Moguće kombinacije se računaju ili traže u tablici na osnovi broja „traga“ i broja slobodnih polja u redu ili stupcu. Često se koristi tablica tzv. „Magičnih“ brojeva. To su brojevi koji u određenom broju polja imaju samo jednu moguću kombinaciju.

Tab. 2.1. Tablica koja sadrži nekoliko „magičnih brojeva“.

Zbroj	Broj kvadrata	Kombinacija
3	2	1+2
4	2	1+3
16	2	7+9
17	2	8+9
6	3	1+2+3
7	3	1+2+4
23	3	6+8+9
24	3	7+8+9

2.2 Program

Program nema grafičko sučelje i poprilično je jednostavan za korištenje, prilikom pokretanja od korisnika traži samo ime Kakura.

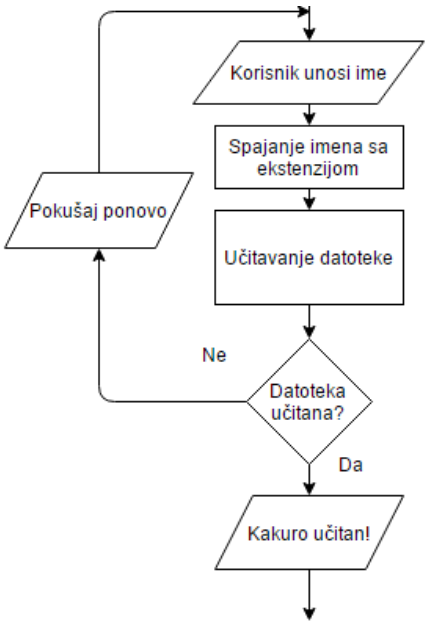
```
Upisi ime kakura kojeg zelis ucitati (bez smjera i ekstenzije npr. 8x8_med_1)
8x8_med_2

Pokusaj ucitavanja 8x8_med_2_dolje.txt i 8x8_med_2_desno.txt je uspio.
Ucitan kakuro!
```

S1. 2.6. Prikaz prilikom unosa

Na osnovu tog imena on pokuša učitati dvije prethodno pripremljene matrice A i B od kojih jedna sadrži vertikalne „tragove“, a druga horizontalne „tragove“. Ukoliko ne pronađe tražene

datoteke, prijavljuje grešku i vraća korisnika nazad na unos. A ukoliko ih pronađe učitava ih i vrši provjeru podataka.



S1. 2.7. Dijagram toka početka programa

Nakon provjere matrice, ovisno o dimenzijama učitanih matrica program određuje dimenzije Kakura. Nema određena ograničenja i učitat će svaki Kakuro.

```
do
{
    t = getc (f);
    if(n==0 && t==' ')
    {
        m++;
    }
    else if(t=='\n')
    {
        n++;
    }

} while (t!= EOF);
```

S1. 2.8. Dio kod-a zadužen za određivanje dimenzija

Nastavlja s učitavanjem samih matrica te stvaranjem tri pomoćna polja: C, D i G. Od kojih C polje služi za popunjavanje i završni prikaz, a D i G za pomoć prilikom rješavanja. C polje je dvodimenzionalno i istih dimenzija kao prethodno učitane tablice. Prilikom stvaranja je podijeljena na „bijela“ i „crna“ polja, a tijekom rada popunjavat će joj se samo „bijela“ polja.

```
int **c = (int **)malloc(n * sizeof(int *));
for (i=0; i<n; i++)
    {
        c[i] = (int *)malloc(m * sizeof(int));
    }
for(i=0; i<n; i++)
    {
        for(j=0; j<m; j++)
        {
            c[i][j]=a[i][j]+b[i][j];
            if(c[i][j]>1){c[i][j]=10;}
        }
    }
```

S1. 2.9. Stvaranje C matrice

D i G polja su trodimenzionalna i omogućavaju rad funkcijama eliminacija i višestruko križanje. Prve dvije dimenzije su iste kao i kod C matrice dok treća sadrži 10 polja koja služe za upis mogućih brojeva. Prvo popunjavanje se odvija tijekom izvršavanja funkcije križanje, u D polje se u svako prazno „bijelo“ polje upisuju svi mogući brojevi koji se mogu nalaziti na tom mjestu.

Odmah potom dolazi do završne petlje koja će ponavljati algoritme dok se ne popuni cijelo polje. Broj ponavljanja ovisi o samim algoritmima.

```
kraj=2;
do{
    krizanje(a,b,c,d);
    zadnjanula(a,c,1);
    zadnjanula(b,c,2);
    prepis(c,d);

    eliminacijadolje(a,b,c,d);
    visestrukokrizanjedolje(a,c,d,g);

    eliminacijadesno(a,b,c,d);
    visestrukokrizanjedesno(b,c,d,g);

    kraj--;
}while(kraj>0);

prikaz(a,b,c);
printf("\n\nKakuro je gotov!\n");
```

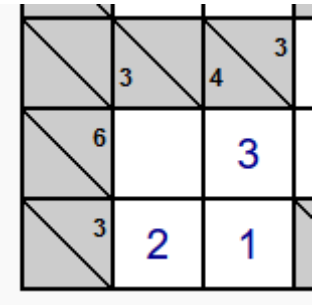
S1. 2.10. Zadnji dio koda

Poziva redom algoritme križanje, zadnja nula, višestruko križanje i eliminacija. Koji koristeći određene pomoćne funkcije postepeno rješavaju Kakuro. Nakon što su se izvršili potreban broj puta program izlazi iz petlje i ispisuje rješenje Kakura.

3. Algoritmi i njihova implementacija

3.1. Zadnja nula

Zadnja nula je jednostavan algoritam koji se koristi ukoliko je u jednom stupcu ili retku ostalo samo jedno prazno „bijelo“ polje, a koristi se nakon što drugi algoritmi popune dovoljan dio tablice.



S1. 3.1. Tehnika zadnja nula

Započinje s pronalaskom reda ili stupca u kojem se nalazi samo jedno prazno polje te nastavlja s zbrajanjem popunjenih „bijelih“ polja tog stupca ili retka. Nakon toga se oduzme zbroj od „traga“ i upisuje razliku u prazno polje. U slučaju slike 3.1. prazno polje se nalazi u stupcu „traga“ 3. U stupcu se nalazi još jedan broj, 2. Oduzimanjem broja 2 od broja 3 dobivamo rješenje, broj 1.

Implementacija

Za izvođenje su mu potrebne matrice „tragova“ A i B te dovoljno popunjena matrica C. Kod započinje s prolaskom kroz matricu A u potrazi za „tragovima“.

```
for(i=0; i<n; i++)
{
    for(j=0; j<m; j++)
    {
        if(a[i][j]>1)
        {
```

S1. 3.2. Pretraga A matrice

Nakon pronalaska prvog „traga“ otvara petlju koja prolazi brojeve u tom stupcu sve dok ne dođe do kraja matrice ili do drugog „traga“. Za vrijeme prolaska zbraja sve brojeve veće od nula te broji nule i pamti položaj zadnje nule.

```
trenbr=a[i][j];
for(k=i+1; k<n; k++)
{
    if(c[k][j]==0)
    {
        brnul++; zadnjik=k;
    }
    else if(c[k][j]>0 && c[k][j]<10)
    {
        zbrojbrojeva=zbrojbrojeva+c[k][j];
    }
    else if(c[k][j]==10)
    {
        break;
    }
}
```

S1. 3.3. petlja koja obavlja pronalazak i zbrajanje

Ako je broj nula nakon izvršavanja petlje jedan znači da se u tom stupcu nalazi samo jedna nula i da je potrebno oduzeti zbroj brojeva od broja „traga“ te upisati u mjesto nule rezultat oduzimanja.

```
if(brnul==1)
{
    c[zadnjik][j]=trenbr-zbrojbrojeva; kraj=2;
}
zbrojbrojeva=0;
brnul=0;
```

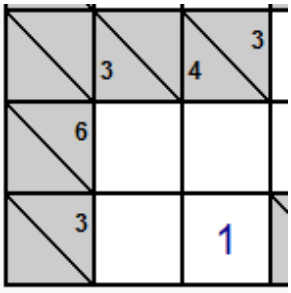
S1. 3.4. završno pitanje zadnje nule

Također, ukoliko se nešto upiše u C matricu postavlja se varijabla kraj na 2 da bi završna petlja znala da je došlo do upisa. Isto se provodi i za B matricu te funkcija završava s radom.

3.2. Križanje

Križanje je algoritam koji ne ovisi o ispunjenosti „bijelih“ polja i po potrebi se koristi u bilo kojem trenutku tijekom rješavanja. Završna petlja započinje s njime i poziva ga cijelo vrijeme tijekom rada. Uz popunjavanje „bijelih“ polja koristi se i za početno ispunjavanje 3D polja. Ima pet postupaka: pronalazak praznog „bijelog“ polja, identifikacija „tragova“, izračun kombinacija, usporedba kombinacija te upis rješenja. Algoritam započinje s pronalaskom praznog „bijelog“ polja u matrici i nastavlja s identifikacijom „tragova“ koji se sijeku na tom polju. Pomoću kojih zajedno s ostalim „bijelim“ poljima tog stupca ili retka računa sve moguće kombinacije brojeva koje se mogu pojaviti u tom stupcu ili retku.

Na kraju vrši usporedbu te dvije skupine kombinacija i ukoliko pronađe samo jedan broj koji se ponavlja upisuje ga.



S1. 3.5. Primjer riješenog polja korištenjem algoritma križanje

Prema primjeru postoji jedna kombinacija za „trag“ 3 i to 1+3 i jedna kombinacija za „trag“ 2, 1+2. Podudara se samo broj 1 i on se upisuje kao rješenje.

Implementacija

Za rad funkcije su potrebne matrice A, B i C. Započinje s prolaskom kroz C matricu u potrazi za nulom.

```
for(i=0; i<n; i++)
{
    for(j=0; j<m; j++)
    {
        if(c[i][j]==0)
        {
```

S1. 3.6. Potraga za nulom

Odmah nakon pronalaska traži „tragove“ koji se sijeku na toj nuli. Prvo „trag“ stupca, a potom trag retka. Traži ih po A i B matricama.

```
k=i;
while(a[k][j]==0)
{
    k--;
}
trbrj=a[k][j];
```

S1. 3.7. Trag stupca

Nakon pronalaska „traga“ otvara petlju koja pronalazi broj nula u stupcu ili retku, oduzima brojeve koji se nalaze u popunjenim „bijelim“ poljima od „traga“ i upisuje te brojeve u

posebno polje. Ti su parametri potrebni da bi pomoćna funkcija kombinacije mogla izračunati listu kombinacija.

```
for(k=k+1; k<n; k++)
{
    if(c[k][j]==0)
    {
        brnul++;
    }
    else if(c[k][j]>0 && c[k][j]<10)
    {
        trbrj=trbrj-c[k][j]; polje[c[k][j]-1]=1;
    }
    else
    {
        break;
```

S1. 3.8. Petlja koja skuplja parametre za funkciju kombinacije

Kada su sakupljeni svi potrebni parametri stvara se dinamički alocirano polje koje služi za spremanje mogućih kombinacija te se sve prosljeđuje pomoćnoj funkciji kombinacije. Koja potom upisuje sve moguće kombinacije u njega. Cijeli postupak se obavlja za oba „traga“ i nastaju dvije matrice popunjene kombinacijama.

```
d=0;
f=(int**)malloc(brnul*sizeof(int*));
kombinacije(brnul,trbrj,f,&d,polje);
```

S1. 3.9. Pozivanje pomoćne funkcije kombinacije

Pomoću četiri for petlje i dva uvjeta vrši se usporedba svih brojeva u matricama kombinacija. Uspoređuje se svaki broj jedne matrice sa svakim brojem druge matrice i kada se pronađe isti zapisuje se da bi se petlja mogla prekinuti ukoliko se pronađe još jedan.

```
for(ik=0; ik<d; ik++)
{
    for(jk=0; jk<brnul; jk++)
    {
        for(ig=0; ig<d2; ig++)
        {
            for(jg=0; jg<brnul2; jg++)
            {
                if(f[jk][ik]==g[jg][ig])
                {
                    if(brojac>0&&g[jg][ig]!=zadnji)
                    {
                        goto izbaci;
                    }
                    zadnji=g[jg][ig];
```

S1. 3.10. Usporedba kombinacija

Ako se petlja izvršila bez izbacivanja znači da postoji samo jedan broj koji se može upisati u to mjesto i on se upisuje.

```
c[i][j]=zadnji; kraj=2;
izbaci:
brojac=0; ukupno=0; zadnji=0;
```

S1. 3.11. Upis rješenja

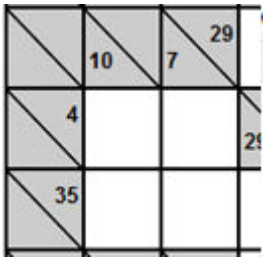
Funkcija nakon izvršavanja algoritma ne prestaje raditi već priprema D polje za daljnji rad tako da upisuje u svako prazno „bijelo“ polje sve moguće brojeve koji se mogu tamo nalaziti.

```
for(ik=0; ik<d; ik++)
{
    for(jk=0; jk<brnul; jk++)
    {
        for(ig=0; ig<d2; ig++)
        {
            for(jg=0; jg<brnul2; jg++)
            {
                if(f[jk][ik]==g[jg][ig])
                {
                    dd[i][j][g[jg][ig]]=g[jg][ig];
```

S1. 3.12. Upis u D polje

3.3. Višestruko križanje

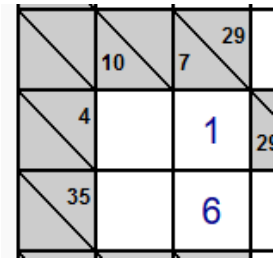
Sama tehnika ili algoritam se mogu koristiti u bilo kojem trenutku rješavanja, čak i na samom početku. Ali u programu se koristi isključivo nakon algoritma križanje zato što koristi D polje koje se moraju prvo popuniti. Zasniva se na izbacivanju kombinacija dok ne ostane samo jedna i ima četiri postupa: odabir „traga“ i izračun kombinacija, prolazak „tragovima“ koji ga sijeku, usporedba kombinacija i izbacivanje te prosljeđivanje rezultata.



S1. 3.13. primjer

Algoritam započinje rad s odabirom „traga“ i izračunom kombinacija za njega. Potom prolazi po redu „tragovima“ koji ga sijeku i uspoređuje kombinacije svakog s kombinacijom odabranog „traga“. Ako se u skupini glavnih kombinacija nalazi kombinacija čiji članovi ne postoje trenutnoj skupini kombinacija, izbacuje se. Ukoliko na kraju izvođenja ostane samo jedna kombinacija veoma lako se može algoritmom križanje dobiti konačno rješenje.

U slučaju slike 3.13. vrši se provjera broja 7. Moguće kombinacije za njega su 1+6, 2+5 i 3+4, a prvi broj koji siječe taj stupac je broj 4 za kojeg su moguće kombinacije 1+3. Što znači da se može izbaciti kombinacija 2+5 pošto se ni jedan broj ne pojavljuje. Drugi broj koji sječe je broj 35. On ima samo jednu kombinaciju i to je 5+6+7+8+9 te na osnovu toga možemo izbaciti kombinaciju 3+4. Sada nam preostaje samo kombinacija 1+6. Nakon toga uz pomoć tehnike križanja lako možemo odrediti koji broj se nalazi u kojem kvadratu.



S1. 3.14. riješeni primjer

Implementacija

Za izvršavanje funkciji su potrebna popunjena polja A, C i D te prazno pomoćno polje G. Započinje rad s traženjem broja upisanog u D polje. Križanje je u prvo mjesto treće dimenzije upisalo broj koji se tamo nalazi, a u ostala brojeve koji se tamo mogu nalaziti.

```
for(i=0; i<n; i++)
{
    for(j=0;j<m;j++)
    {
        for(k=0;k<10;k++)
        {
            if(d[i][j][k]!=0)
            {
```

S1. 3.15. Pronalazak mogućeg broja u D polju

Potom upisuje odabrani broj u polje iskorištenih brojeva i smanjuje broj nula za 1, tj. „upisuje“ taj broj u to prazno mjesto. Nakon toga traži „trag“ kojem pripada taj broj i odmah

ga oduzima od njega. If usporedbom provjerava da li je broj već na to mjestu upisan da ne prestane s radom nakon kasnije nakon ponovne provjere.

```
polje[(d[i][j][k])-1]=1;
brnul=-1;
k2=i;
while(a[k2][j]==0)
{
    k2--;
}
trbrj=a[k2][j];
trbrj=trbrj-d[i][j][k];
if(d[i][j][k]==c[i][j])
{
    nepreskok=1;
    trbrj=trbrj+c[i][j];
    brnul++;
}
poc=k2+1;
```

S1. 3.16. Pronalazak „traga“

Kada ima „trag“ kreće s traženjem broja praznih „bijelih“ polja, oduzimanjem trenutnih popunjenih od „traga“ te upisivanjem iskorištenih brojeva u predviđeno polje. Prilikom zapisa popunjenog „bijelog“ broja uspoređuje se s početnim odabranim brojem i ukoliko su jednaki, a nije uneseno da se taj broj nalazi u C matrici broj se preskače i ide se na idući. Isto se događa i ako nema slobodnih nula.

```
for(k2=k2+1; k2<n; k2++)
{
    if(c[k2][j]==0)
    {
        brnul++;
    }
    else if(c[k2][j]>0 && c[k2][j]<10)
    {
        if(d[i][j][k]==c[k2][j]&&nepreskok==0)
        {
            goto skok;
        }
        trbrj=trbrj-c[k2][j];
        polje[c[k2][j]-1]=1;
    }
    else{break;}
}
```

S1. 3.17. Pregled „bijelih“ polja tog stupca

Na osnovu broja nula, konačnog „traga“ i polja upisanih brojeva stvara se polje kombinacija pomoću funkcije kombinacije. Potom se ponovo prolazi stupcem i na mjestu svakog praznog

„bijelog“ polja uspoređuju se mogući brojevi s svim mogućim kombinacijama brojeva. Prilikom podudaranja broj se upisuje u pomoćno G polje. A nakon završetka funkcije poziva se pomoćna funkcija prijepis koja čisti treću dimenziju dijela D polja u kojoj je došlo do promjene i upisuje elemente iz G polja. Na kraju se pomoćnom funkcijom čišćenje čisti cijelo G polje.

```
for(k2=poc; k2<n; k2++)
{
    if(c[k2][j]==0)
    {
        for(ig=0; ig<10; ig++)
        {
            for(ik=0; ik<d2; ik++)
            {
                for(jk=0; jk<brnul; jk++)
                {
                    if(d[k2][j][ig]==f[jk][ik])
                    {
                        g[k2][j][ig]=f[jk][ik];
                    }
                }
            }
        }
    }
    else if(c[k2][j]==10)
    {
        break;
    }
}
```

S1. 3.18. Upis podudaranih brojeva u G polje

Na samom kraju izvođenja funkcije poziva se zadnja pomoćna funkcija koja provjerava cijelu C matricu i ukoliko pronađe polje gdje se nalazi samo jedan mogući broj upisuje ga kao rješenje.

3.4. Eliminacija

Poput križanja i višestrukog križanja tehnika eliminacije se može koristiti u bilo kojem trenutku tijekom rješavanja Kakura. Ona se zasniva na pronalasku mogućnosti postavljanju istih brojeva u isti broj različitih polja u jednom retku ili stupcu. Započinje s pronalaskom reda u kojem se pojavljuju dva ili više praznih polja koja mogu sadržavati iste brojeve. „Zamišlja“ se da su ti brojevi upisani u ta polja i pokušava se riješiti ostatak retka ili stupca korištenjem neke druge tehnike.

Implementacija

Za rad uzima polja A, C i D, a započinje s traženjem prvog praznog polja u C matrici i određivanjem „traga“ uz pomoć A matrice.

```
for(i=0; i<n; i++)
{
    for(j=0; j<m; j++)
    {
        if(c[i][j]==0)
        {
            k3=i;
            while(a[k3][j]==0)
            {
                k3--;
            }
```

S1. 3.19. Početak eliminacije

Nastavlja s brojanjem praznih „bijelih“ polja u tom stupcu ili retku. Te upisuje brojeve koji se nalaze u odabranom polju u brojač i broji ih.

```
for(h=k4; h<n; h++)
{
    if(c[h][j]==0)
    {
        brnul++;
    }
    else if(c[h][j]==10)
    {
        break;
    }
}
for(h=1; h<10; h++)
{
    if(d[i][j][h]!=0)
    {
        brojac[d[i][j][h]]=d[i][j][h];
        brojac1++;
    }
}
```

S1. 3.20. Brojanje

Potom ponovo prolazi kroz taj stupac ili redak i uspoređuje sve moguće brojeve svakog praznog „bijelog“ polja s brojevima prvog polja i zapisuje broj podudaranja.

```
for(k2=k4; k2<n; k2++)
{
    if(c[k2][j]==0)
    {
        for(h2=1; h2<10; h2++)
        {
            if(brojac[h2]==d[k2][j][h2])
            {
                brojac2++;
            }
        }
        if(brojac2==9)
        {
            brojac3++;
        }
        brojac2=0;
    }
    else if(c[k2][j]==10)
    {
        break;
    }
}
```

S1. 3.21. Provjera brojeva

Nakon prolaska svih praznih „bijelih“ polja funkcija dolazi do usporedbe i provjerava se da li je broj brojeva prvog polja jednak broju polja u kojima se oni podudaraju te da li je broj tih polja manji od nula. Ako su uvjeti zadovoljeni ulazi se u glavnu petlju u kojoj se ponovo ide po svim praznim „bijelim“ poljima tog retka ili stupca. U njoj se vrši usporedba trenutnog polja s početnim poljem te ukoliko nisu isti izbacuju se brojevi koji se nalaze u početnom polju.

```
if(brojac1==brojac3 && brojac1<(brnul))
{
    for(h2=k4; h2<n; h2++)
    {
        if(c[h2][j]==0)
        {
            for(h=1; h<10; h++)
            {
                if(d[h2][j][h]==brojac[h] && brojac[h]>0)
                {
                    brojacbr++;
                }
                else if(d[h2][j][h]>0)
                {
                    brojacbr2++;
                }
            }
            if(brojac1==brojacbr && brojacbr2>0)
            {
                for(h=1; h<10; h++)
                {
                    if(brojac[h]==d[h2][j][h] && brojac[h]!=0)
                    {
                        d[h2][j][h]=0;
                    }
                }
            }
                    brojacbr=0;
                    brojacbr2=0;
```

S1. 3.22. Eliminacija brojeva

Nakon izvršavanja algoritma funkcija još poziva pomoćnu funkciju provjere koja upisuje rješenja ukoliko ih nađe u C matricu.

4. Pomoćne funkcije

4.1 Kombinacije

Svrha kombinacija je da na osnovu primljenog glavnog broja ili „traga“, broja slobodnih polja i iskorištenih brojeva vrača matricu u kojoj se nalaze moguće kombinacije brojeva za ta polja.

Npr. Ukoliko primi broj 20 kao glavni broj, broj tri kao broj slobodnih kvadrata i ništa za iskorištene brojeve vratit će matricu koja sadržava kombinacije 3+8+9, 4+7+9, 5+6+9 i 5+7+8. No ukoliko primi isti glavni broj, isti broj slobodnih kvadrata ali i broj 7 kao iskorišteni broj, vratit će samo kombinacije 4+7+9 i 5+7+8.

Koriste ju svi algoritmi, ona je jedna od najvažnijih funkcija i zadužena je za svu kombinatoriku u programu. Iako ju samo algoritam križanje poziva direktno ostali ju koriste preko trodimenzionalnih polja. Sastoji se od tri funkcije: kombinacije, broj i dodaj red.

4.1.1 Funkcija kombinacije

Pomoćna funkcija koja je napravljena da bi se pojednostavnilo pozivanje ostalih funkcija. Prima glavni broj, broj slobodnih kvadrata, dinamički alocirano polje, broj d i polje iskorištenih brojeva. Potom stvara pomoćno polje n koje ima broj elemenata jednak broju nula i prosljeđuje ga zajedno s primljenim podatcima funkciji br.

```
void kombinacije(int a,int b,int **c,int *d,int *e)
{
    int *n;
    int i;
    n=(int*)malloc(a*sizeof(int));
    br(a,a,b,n,1,c,d,e);
    free(n);
}
```

S1. 4.1. Funkcija kombinacije

4.1.2 Funkcija broj

Funkcija broj je rekurzivna funkcija, tj. funkcija koja poziva samu sebe. Prima vrijednosti od funkcije kombinacije koje su: broj slobodnih kvadrata, broj slobodnih kvadrata, glavni broj , polje n, broj jedan, dinamički alociranu matricu, broj redaka matrice i polje sa iskorištenim brojevima

Ovisno o broju nula koji je primila započinje rad na jedan od dva načina. Ukoliko primi broj jedan ili dođe do njega tijekom izvođenja započinje s prvim koji provjerava da li je glavni broj manji ili jednak devet, da li je veći ili jednak trenutnom broju i da li je već iskorišten. Ako je manji ili jednak devet, veći je od trenutnog broja i nije iskorišten upisuje se u prvo mjesto pomoćnog n polja. Nakon toga se poziva funkcija dodaj red kojoj se šalju dinamički alocirano polje, broj stupca i broj nula. Odmah nakon izvršavanja upisuje elemente iz pomoćne n matrice u glavnu matricu.

```
if(b<=9&&b>=j&&e[b-1]!=1)
{
    n[0]=b;
    dodajred(h,d,c);
    for(i=c-1;i>=0;i--)
    {
        h[i][*d-1]=n[i];
    }
}
```

S1. 4.2. Prvi slučaj

Vrlo često neće primiti broj jedan i tada započinje petlju koja broji i cijelo vrijeme provjerava da li je razlika između glavnog broja i trenutnog broja veća od nula. Odmah nakon provjerava da li je trenutni broj već iskorišten i ukoliko je preskače ga, a ukoliko nije upisuje u pomoćnu matricu n. Funkcija potom ponovo poziva sama sebe, ali vrši određene izmjene. Smanjuje broj nula za jedan i glavni broj za trenutni broj te povećava trenutni broj za jedan.

```
for(i=j;i<=10-a;i++)
{
    if(b-i>0)
    {
        if(e[i-1]==1)continue;
        n[a-1]=i;
        br(c,a-1,b-i,n,i+1,h,d,e);
    }
    else
    {
        break;
    }
}
```

S1. 4.3. Drugi slučaj

4.1.3 Funkcija dodaj red

Izvršava alokaciju ili realokaciju stupaca glavne matrice ovisno. A prima trenutnu matricu sa kombinacijama, njezin broj redaka i broj stupaca.

```
int i,j;
if(*d==0)
{
    for(i=0;i<a;i++)
    {
        c[i]=(int*)malloc(8*sizeof(int));
    }
    (*d)++;
}
else if((*d)%8==0)
{

    j=*d*sizeof(int);
    for(i=0;i<a;i++)
    {
        c[i]=(int*)realloc(c[i],j);
    }
    (*d)++;
}
else (*d)++;
```

S1. 4.4. Funkcija dodaj red

Prvo provjerava broj stupaca te ako je jednak nuli ulazi u prvu petlju koja alocira osam stupaca glavnoj matrici te to i zabilježi. Ako je broj stupaca višekratnik broja osam pokreće se realokacija matrice i dodaje joj se još osam stupaca. A ukoliko ne zadovoljava ni jedan od ta dva uvjeta samo se povećava varijabla d za jedan, tj. signalizira upis u idući stupac.

4.2 3D polja

3D polja je skupina od pet funkcija: prebacivanje G u D, čišćenje 3D, oslobodi 3D, prijepis i provjera. Zadužene su za prijenos brojeva između C, D i G polja, čišćenje i oslobađanje memorije D i G polja te završnu provjeru i unos konačnog broja u C polje. Funkcija prijepis se koristi iza algoritama zadnja nula i križanje, a ostale se pozivaju direktno iz algoritama višestruko križanje i eliminacija.

4.2.1 Funkcija prijepis

Jednostavna funkcija koja upisuje sadržaj C polja u D polje, tj. priprema D polje za obradu. Prima dvije varijable, a to su C i D polja.

```
void prijepis(int **c, int ***d)
{
    int i,j,k;
    for(i=0; i<n; i++)
    {
        for(j=0;j<m;j++)
        {
            if(c[i][j]>0 && c[i][j]<10)
            {
                for(k=0;k<10;k++)
                {
                    d[i][j][k]=0;
                }
                d[i][j][0]=c[i][j];
```

S1. 4.5. Funkcija prijepis

Provjerava postojanje broja u C matrici i ako postoji čisti treću dimenziju D polja te upisuje sadržaj C matrice u prvo mjesto.

4.2.2 Funkcija prebacivanje G u D

Služi za preslikavanje pomoćnog G polja u D polje. Prima dvije varijable (u ovom slučaju D i G polja) i prepisuje sadržaj iz jednog polja u drugo.

```
void prebacivanjegud(int ***a, int ***b)
{
    int i,j,k;
    for(i=0;i<n;i++)
    {
        for(j=0;j<m;j++)
        {
            for(k=0;k<10;k++)
            {
                a[i][j][k]=b[i][j][k];
```

S1. 4.6. Funkcija prebacivanje g u d

4.2.3 Funkcija čišćenje 3D

Funkcija koja prima samo jedno 3D polje kao varijablu i mijenja sve njezine elemente s nulom. Putuje po polju i jednu po jednu vrijednost mijenja s nulom.

```
void ciscenje3d(int ***a)
{
    int i,j,k;
    for(i=0;i<n;i++)
    {
        for(j=0;j<m;j++)
        {
            for(k=0;k<10;k++)
            {
                a[i][j][k]=0;
            }
        }
    }
}
```

S1. 4.7. Funkcija čišćenje 3 D

4.2.4 Funkcija oslobodi 3D

Pošto su sva polja dinamički alocirana potrebno je nakon kraja njihovog korištenja osloboditi zauzeti prostor. Funkcija oslobodi 3D to obavlja za trodimenzionalna polja. Prima dinamički alocirano polje i njegove dimenzije te uz pomoć dvije for petlje čisti memoriju.

```
void oslobodi3d(int ***a)
{
    int i,j;
    for(i=0;i<n;i++)
    {
        for(j=0;j<m;j++)
        {
            free(a[i][j]);
        }
        free(a[i]);
    }
    free(a);
}
```

S1. 4.8. Funkcija oslobodi 3D

4.2.5 Funkcija provjera

Obavlja završnu provjeru D polja i upisivanje rezultata u C matricu. Izvodi se iza algoritama eliminacija i višestruko križanje. Provjerava da li je u nekom od polja D polja ostao samo

jedan broj. Ovisno o rezultatu preskače ili upisuje taj broj u C matricu. Ako je došlo do upisa čisti to polje D polja.

```
void provjera(int ***a, int **b)
{
    int i,j,k,brojac=0,zadnji,reset;
    for(i=0;i<n;i++)
    {
        for(j=0;j<m;j++)
        {
            if(b[i][j]==0)
            {
                for(k=0;k<10;k++)
                {
                    if(a[i][j][k]>0)
                    {
                        brojac++; zadnji=a[i][j][k];
                    }
                }
                if(brojac==1)
                {
                    b[i][j]=zadnji;
                    for(k=0;k<10;k++)
                    {
                        a[i][j][k]=0;
                    }
                    kraj=2;
                }
                brojac=0;
                zadnji=0;
```

S1. 4.9. Funkcija provjera

4.3 Ostale funkcije

Ovdje su navedene i opisane preostale dvije pomoćne funkcije programa. To su prikaz i oslobodi 2D.

4.3.1 Funkcija prikaz

Nalazi se na kraju programa i služi za jednostavan prikaz Kakuro matrice. „Bijela“ polja su ispunjena, a „crna“ su označena ovisno o sadržaju. Ako je „crno“ polje prazno označeno je s znakom x, a ako ima jedan ili dva broja u sebi postavlja ih na lijevu ili desnu stranu ovisno o pravcu djelovanja.

```
  x     5\x   17\x    x     x    11\x   23\x   21\x   15\x

 x\9    1     8     10\x  27\26  9      8      2      7

 x\42   4     9     7     5      2      6      1      8

  x     8\x   24\9   3     6     12\15  9      6      4\x

 x\15   7     8     x\9   7      2     x\12   9      3

 x\5    1     4     20\13 9      4     16\4   3      1

  x    14\16  7     9     17\14  5     9      10\x   15\x

 x\37   5     3     4     9      1     7      2      6

 x\26   9     2     7     8      x     x\17   8      9
```

S1. 4.10. Ispis riješenog kakura

Potrebne varijable su matrice A, B i C. Sastoji se od dvije for petlje i zasniva se na usporedbi matrica.

```
void prikaz(int **a, int **b, int **c)
{
    int i,j,k;
    for(i=0; i<n; i++)
    {
        printf("\n\n");
        for(j=0; j<m; j++)
        {
            if(c[i][j]<10)
            {
                printf("   %d   ",c[i][j]);
            }
            else if(a[i][j]==1 && b[i][j]==1)
            {
                printf("   x   ");
            }
```

S1. 4.11. Dio funkcije prikaz

4.3.2 Funkcija oslobodi 2D

Ima istu ulogu kao i funkcija oslobodi 3D, ali ova to obavlja za dvodimenzionalna polja. Prima dinamički alocirano polje i njegove dimenzije te oslobađa zauzetu memoriju. Izvršava se nakon svakog izvršavanja kombinacija te na samom kraju programa.

```
void oslobodi(int **c,int a)
{
    while(a>0)
    {
        free(c[--a]);
    }
    free(c);
}
```

S1. 4.12. Funkcija oslobodi 2D

5. Zaključak

Zadatak ovog rada je bio analiziranje logičke igre Kakuro i tehnika korištenih za rješavanje te razrađivanje tih tehnika i implementiranje uz pomoć programskog jezika C.

Preko analize četiri različite tehnike rješavanja se određivao i opisivao njihov način rada. Na kraju analize je vidljivo da svaka od četiri različite tehnike rješavanja ne može samostalno riješiti igru te se svaka primjenjuje u točno određenom trenutku.

Sve tehnike rješavanja Kakura koriste kombinaciju kombinatorike i logike. U radu je opisana svaka tehnika zasebno iz kojih je vidljivo da svaka rješava problem Kakura na svoj jedinstveni način te se također zaključuje da niti jedna od tih tehnika nije pogrešna jer sve dolaze do točnog rješenja.

Prilikom implementacije, bilo je potrebno izmjenjivati algoritme da bi ih računalo moglo izvršiti, iako se baziraju na logici i kombinatorici. To je bilo potrebno zato što su tehnike osmišljene za ljudsku upotrebu.

Tijekom rada na programu, bilo je potrebno uvesti mnoštvo pomoćnih funkcija koje nisu direktno povezane sa samim tehnikama rješavanja. Bez pomoćnih funkcija, računalo ne bi moglo prikazati ni riješiti Kakuro.

LITERATURA

1. Timmerman Charles, The Everything Kakuro Challenge Book , https://en.wikipedia.org/wiki/Kakuro, pristup ostvaren 08.06.2016.

2. What is Kakuro. http://www.kakurolive.com/about-kakuro.php, pristup ostvaren 11.06.2016.

3. How Kakuro Puzzles Work, http://entertainment.howstuffworks.com/puzzles/kakuro-puzzles1.html, pristup ostvaren 11.06.2016.

4. Kakuro solving techniques, http://www.kakuro.com/techniques.php, pristup ostvaren 12.06.2016.

5. Solving Kakuro, http://rohanrao.blogspot.hr/2011/01/solving-kakuro-unique-combinations.html, pristup ostvaren 16.06.2016.

SAŽETAK/SUMMARY

SAŽETAK

Zadatak rada je proučavanje igre i tehnika rješavanja te njihova razrada i implementacija uz pomoć programskog jezika C.
Rad započinje sa definicijom, opisom pravila i povijesti logičke igre Kakuro. Uz veliku važnost logike, spominje se i kombinatorika te se opisuju tehnike rješavanja.
Vrši se dodatna razrada tehnika, prikazuje se način rada na primjerima te se opisuje način implementacije tehnika u programu. Prikazuje se razlika izvođenja algoritma između ljudi i računala.
Na kraju rada opisane su pomoćne funkcije dodane za omogućavanje izvođenja algoritama računalu. Prikazana je njihova velika važnost, iako nisu direktno povezane sa tehnikama rješavanja.

Ključne riječi: Igra, Tehnike rješavanja, Implementacija, Definicija, Logička igra, Kakuro, Logika, Kombinatorika, Program, Algoritam, Funkcija.

KAKURO IN PROGRAMMING LANGUAGE C

SUMMARY

Point of the work is to study the game, its solving techniques and implementation of solving techniques by the use of C language
Paper starts with the definition of game, description of rules and history of Kakuro. Aside from great importance of logic, combinatorics is mentioned and solving techniques are described.
Additional elaboration of techniques is performed, work performance is shown on examples and technique implementation is described. Difference in performance of algorithm between people and computers is shown.
Support functions are described at the end of the paper and although they are not directly linked to solving techniques, their importance is shown.

Keywords: Game, Solving techniques, Implementation, Definition, Logic game, Kakuro, Combinatorics, Program, Algorithm, Function.

Životopis

Ivan Maričević rođen je 22.02.1993. Živio je u Požegi i tamo pohađao osnovnu školu Antuna Kanižlića. Po završetku upisuje Tehničku školu smjer tehničar za mehatroniku također u Požegi.

Tijekom školovanja bio je zainteresiran za matematiku i engleski te išao na dodatne sate i natjecanja.

U slobodno vrijeme bavio se video igrama, streljaštvom, gimnastikom i streličarstvom te je sudjelovao u većem broju turnira.

Nakon srednje škole nastavlja školovanje na Elektrotehničkom fakultetu u Osijeku, upisao je stručni studij elektrotehnike smjer informatika.