



## **Dokumentacija Projektnog Zadatka A31**

Pogađanje broja

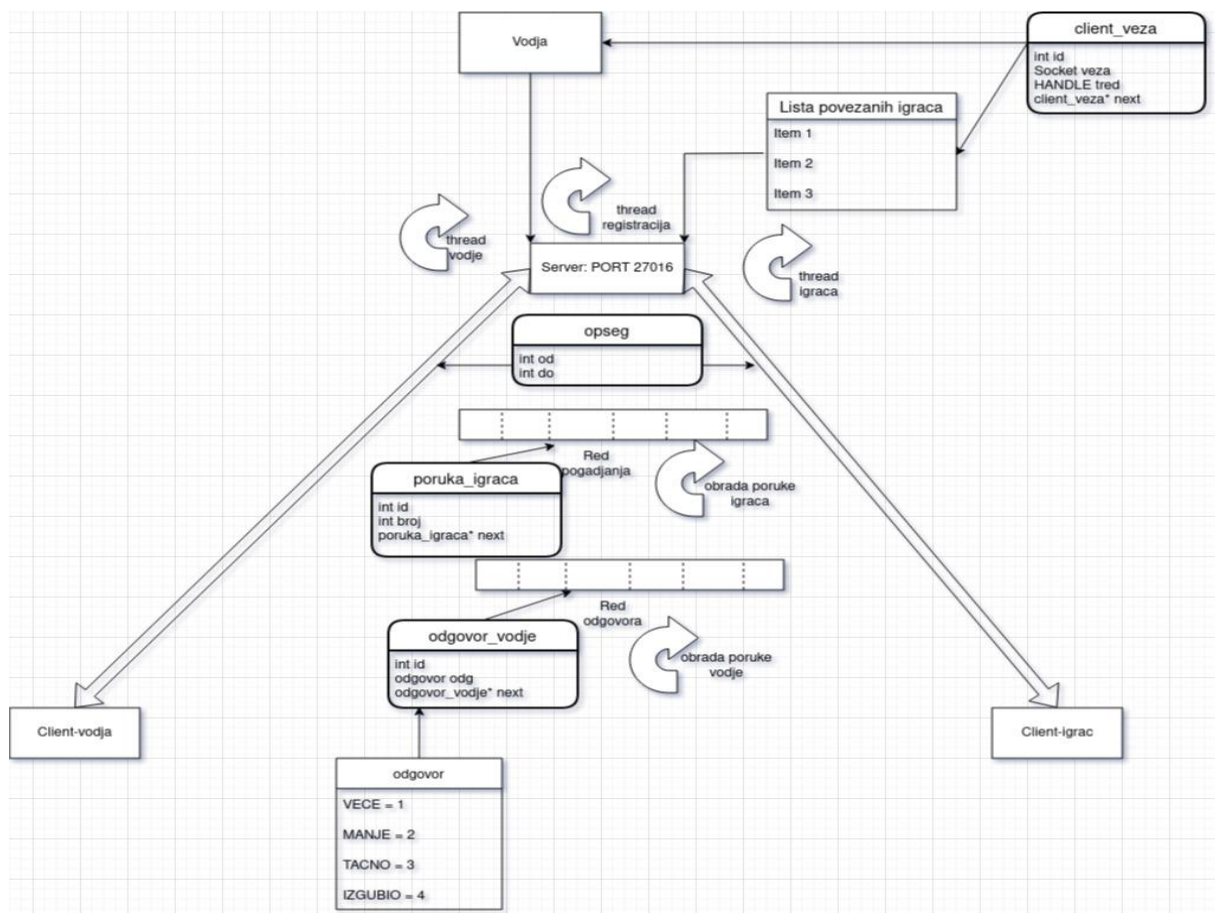
# Sadržaj

Uvod .....	3
Opis dizajna .....	3
Server .....	4
Client .....	7
Strukture podataka .....	9
Rezultati testiranja .....	11
Zaključak.....	13

## Uvod:

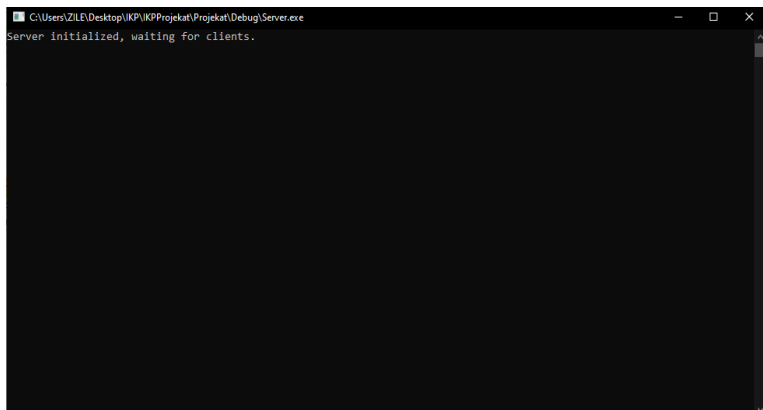
Potrebno je napraviti igru pogađanja zamisljenog broja koja se sastoji od dve komponente a to su server i neograničen broj klijenata. Cilj zadatka je omogućiti komunikaciju servera i klijenata, da obezbedimo odabir uloge igrača i vođe, kao i da realizujemo logiku pogađanja broja gde klijent sam može da bira da li će pogađati ručno ili uz pomoć algoritma.

## Opis dizajna:



Slika 1

**Server**- prva uloga mu je da osluškuje kanal i prihvata konekcije na PORT-u 27016. Prihvat konekcije radi tako sto Server pravi



Slika 2

thread(thread registracija na slici) koji služi za kreiranje klijentskog socketa i pravljenja klijentskog threada(thread vodje/thread igraca na

slici).

Pored thread za prihvat Server pravi dva nova thread-a, prvi koji je namenjen za citanje poruka igrača sa reda (obrada poruke igraca na slici) i drugi koji služi za citanje poruka vođe sa reda (obrada poruke vođe na slici).

```
DWORD WINAPI slanje_vodji(LPVOID param)
{
    while (1) {
        WaitForSingleObject(slanje_vodji_sem, INFINITE);
        EnterCriticalSection(&cs);
        do {
            poruka_igraca* poruka = red_poruka_igraca_out;
            red_poruka_igraca_out = poruka->next;
            int iResult = send(vodja_igre->socket, (const char*)poruka, sizeof(poruka_igraca), 0);
            if (iResult == SOCKET_ERROR)
            {
                printf("send failed with error: %d\n", WSAGetLastError());
                closesocket(vodja_igre->socket);
                free(vodja_igre);
                vodja_igre = NULL;
            }

            free(poruka);
        } while (red_poruka_igraca_out != NULL);
        red_poruka_igraca_in = NULL;
        poruka_igraca poruka;
        poruka.id = -1;
        int iResult = send(vodja_igre->socket, (const char*)&poruka, sizeof(poruka_igraca), 0);
        if (iResult == SOCKET_ERROR)
        {
            printf("send failed with error: %d\n", WSAGetLastError());
            closesocket(vodja_igre->socket);
            free(vodja_igre);
            vodja_igre = NULL;
        }

        LeaveCriticalSection(&cs);
    }
}
```

Slika 3

Thread obrada poruke igrača ima ulogu da preuzima poruke sa reda(red pogađanja na slici) i šalje klijentu koji ima ulogu vođe,a uloga thread-a obrada poruke vođe je da preuzme poruke sa reda(red odgovora na slici),provera da li je detektovan pobednik i salje poruke odgovarajućim klijentima.U ovim thread-ovima korišćeni su semafori čija uloga je bila da sačekaju smeštanje svih poruka na red i nakon toga izvršavanje gore napisane logike.

```
DWORD WINAPI slanje_igracima(LPVOID param) {  
    while (1) {  
        WaitForSingleObject(slanje_igracima_sem, INFINITE);  
        odgovor_vodje* t = red_poruka_vodje_out;  
  
        do {  
            EnterCriticalSection(&cs);  
            odgovor_vodje* odg = red_poruka_vodje_out;  
            if (odg->id != -1) {  
                client_veza* head = povezani_igraci;  
                while (head->id != odg->id) {  
                    head = head->next;  
                }  
                int buf = (int)odg->odg;  
                if (pobednik_id != -1) {  
                    if (pobednik_id != odg->id) {  
                        buf = IZGUBIO;  
                    }  
                }  
                int res = send(head->socket, (const char*)&buf, sizeof(int), 0);  
                if (res == SOCKET_ERROR) {  
                    printf("GRESKA : %d\n", WSAGetLastError());  
                    closesocket(head->socket);  
                    oslobodi_klijenta(head);  
                }  
            }  
  
            red_poruka_vodje_out = red_poruka_vodje_out->next;  
            free(odg);  
  
            LeaveCriticalSection(&cs);  
        } while (red_poruka_vodje_out != NULL);  
  
        red_poruka_vodje_in = NULL;  
    }  
}
```

Slika 4

Klijentski thread je zadužen za registraciju običnih igrača i vođe, ukoliko dođe do pokušaja registracije vođe koji već postoji taj klijent će biti registrovan kao igrač. U zavisnosti od toga da li je klijent registrovan kao vođa ili kao igrač thread će izvršavati dve odvojene logike. Kada je klijent registrovan kao igrač njegov thread će prihvatiti poruku u smestiti u red (red pogađanja na slici).

```
poruka_igraca* tmp = (poruka_igraca*)malloc(sizeof(poruka_igraca));
int broj = 0;
iResult = recv(client.socket, (char*)&broj, sizeof(int), 0);

if (iResult > 0) {
    EnterCriticalSection(&cs);
    tmp->broj = broj;
    tmp->id = client.id;
    tmp->next = NULL;
    broj_poslatih_poruka_igraca++;
    if (red_poruka_igraca_in == NULL)
    {
        red_poruka_igraca_in = tmp;
        red_poruka_igraca_out = tmp;
    }
    else
    {
        red_poruka_igraca_in->next = tmp;
        red_poruka_igraca_in = tmp;
    }
    LeaveCriticalSection(&cs);
    printf("Broj pristiglih poruka %d\n", broj_poslatih_poruka_igraca);
    if (broj_igraca == broj_poslatih_poruka_igraca)
    {
        broj_poslatih_poruka_igraca = 0;
        ReleaseSemaphore(slanje_vodji_sem, 1, NULL);
    }
}
```

Slika 5

Ako je klijent registrovan kao vođa njegov thread će prvo prihvatiti opseg koji će biti prosleđen svim klijentima i nakon toga gasi thread za prihvatanje konekcije, posle toga prihvata poruke i smesta u red (red odgovora na slici). Nakon pogađanja tačnog broja ponovo se pali thread za prihvatanje konekcije i može se ponoviti igra.

```

if (!rcvCnt) {

    iResult = recv(client.socket, (char*)&op, sizeof(op), 0);
    if (iResult > 0) {

        printf("%d - %d\n", op.od, op.doo);
        PocetakIgre(op);

        TerminateThread(prihvatanje_konekcije_tr, 0);
        rcvCnt++;
    }
    else if (iResult == 0) {
        printf("Vodja prekinuo konekciju\n");
        closesocket(vodja_igre->socket);
        free(vodja_igre);
        vodja_igre = NULL;
        kraj = 0;
    }
    else {
        printf("Dostupno je do greske prilikom prijema na vodji. Error %d\n", WSAGetLastError());
        closesocket(vodja_igre->socket);
        free(vodja_igre);
        vodja_igre = NULL;
        kraj = 0;
    }
}

```

Slika 6

```

else {
    odgovor_vodje* odg;
    do {
        odg = (odgovor_vodje*)malloc(sizeof(odgovor_vodje));
        iResult = recv(client.socket, (char*)&odg, sizeof(odgovor_vodje), 0);

        if (iResult > 0) {
            if (odg->odg == TACNO) {

                if (pobednik_id == -1) {
                    pobednik_id = odg->id;
                }
            }

            odg->next = NULL;
            if (odg->id != -1) {
                EnterCriticalSection(&cs);

                if (red_poruka_vodje_in == NULL) {

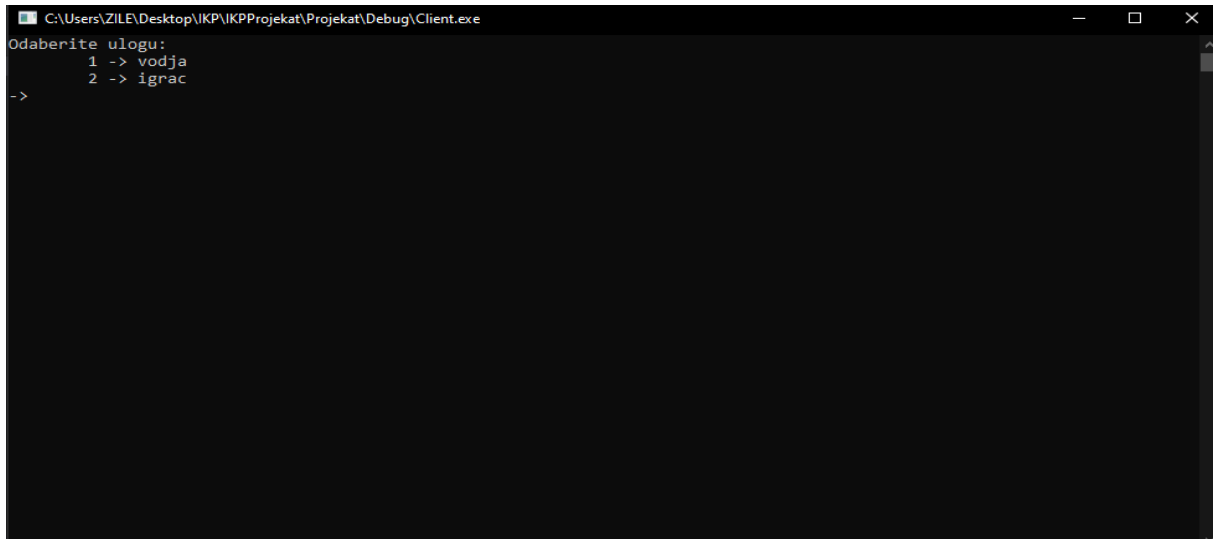
                    red_poruka_vodje_in = odg;
                    red_poruka_vodje_out = odg;
                }
                else {

                    red_poruka_vodje_in->next = odg;
                    red_poruka_vodje_in = odg;
                }
                LeaveCriticalSection(&cs);
            }
        }
    }
}

```

Slika 7

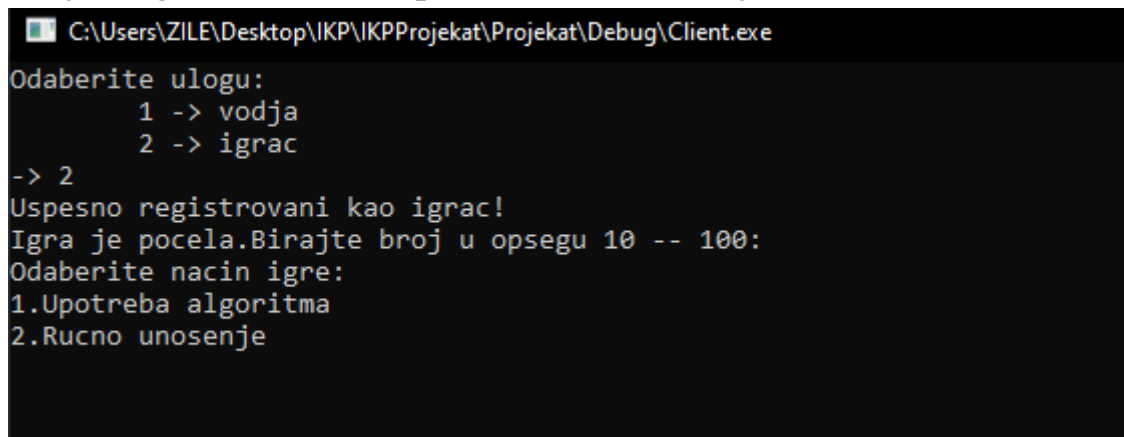
**Client**-prvi zadatak mu je povezivanje sa Server-om na Port-u 27016. Nakon uspešne konekcije bira ulogu(vođa/igrač). Kada se svi igrači konektuju i igrač koji je povezan kao vođa pošalje opseg, igra počinje.



Slika 8

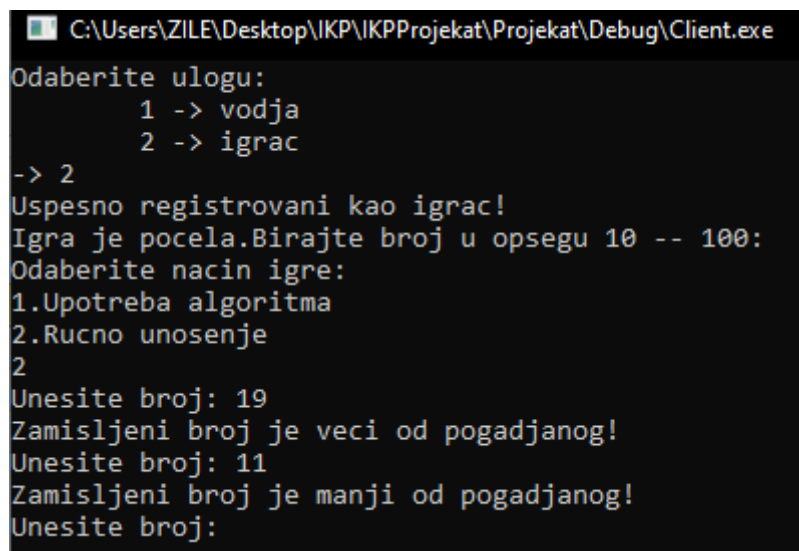
Ako je klijent povezan kao igrač prihvata opseg i bira da li će pogađati broj ručno ili uz pomoć algoritma. Nakon što su svi klijenti poslali

svoje odgovore čeka se povratna informacija od servera.



```
C:\Users\ZILE\Desktop\IKP\IKPProjekat\Projekat\Debug\Client.exe
Odaberite ulogu:
    1 -> vodja
    2 -> igrac
-> 2
Uspesno registrovani kao igrac!
Igra je pocela.Birajte broj u opsegu 10 -- 100:
Odaberite nacin igre:
1.Upotreba algoritma
2.Rucno unosenje
```

Slika 9



```
C:\Users\ZILE\Desktop\IKP\IKPProjekat\Projekat\Debug\Client.exe
Odaberite ulogu:
    1 -> vodja
    2 -> igrac
-> 2
Uspesno registrovani kao igrac!
Igra je pocela.Birajte broj u opsegu 10 -- 100:
Odaberite nacin igre:
1.Upotreba algoritma
2.Rucno unosenje
2
Unesite broj: 19
Zamisljeni broj je veci od pogadjanog!
Unesite broj: 11
Zamisljeni broj je manji od pogadjanog!
Unesite broj:
```

Slika 10

Prilikom prihvata povratne informacije proverava se vrednost koja je stigla od servera, ako je stiglo „MANJE/VEĆE“ korisnik nastavlja sa pogađanjem.

Ukoliko je stiglo „IZGUBIO“ to nam govori da je neko od klijenata pogodio zamišljen broj i igra se završava a ako je klijent primio „TAČNO“ to nam govori da je taj klijent pogodio i da je on pobednik igre.



```
C:\Users\ZILE\Desktop\IKP\IKPProjekat\Projekat\Debug\Client.exe
Odaberite ulogu:
    1 -> vodja
    2 -> igrac
-> 2
Uspesno registrovani kao igrac!
Igra je pocela.Birajte broj u opsegu 10 -- 100:
Odaberite nacin igre:
1.Upotreba algoritma
2.Rucno unosenje
2
Unesite broj: 15
Zamisljeni broj i broj koji ste poslali se podudaraju!
Pritisni dugme za gasenje programa !
```

Slika 11

```
C:\Users\ZILE\Desktop\IKP\IKPProjekat\Projekat\Debug\Client.exe
Odaberite ulogu:
    1 -> vodja
    2 -> igrac
-> 2
Uspesno registrovani kao igrac!
Igra je pocela.Birajte broj u opsegu 10 -- 100:
Odaberite nacin igre:
1.Upotreba algoritma
2.Rucno unosenje
2
Unesite broj: 19
Igra je završena,izgubili ste!
Pritisni dugme za gasenje programa !
```

Slika 12

Ako je klijent povezan kao vođa,prvo unosi broj koji je zamislio,nakon čega salje opseg na server čime označava početak igre. Posle nekog vremena prelazi se u fazu prijema poruka sa servera i provere brojeva.Vođa proverava zamišljeni broj sa primljenim brojevima i u zavisnosti od toga da li je broj veći,manji ili mozda tačan pakuje odgovore i šalje nazad serveru.Ukoliko je neki od odgovora tačan igra se završava i vođa gasi konekciju sa serverom.

```
C:\Users\ZILE\Desktop\IKP\IKPProjekat\Projekat\Debug\Client.exe
Odaberite ulogu:
    1 -> vodja
    2 -> igrac
-> 1
Uspesno registrovani kao vodja!
Unesite zamisljen broj: 15
Unesite opseg brojeva!
Od: 10
Do: 100
```

Slika 13

```
C:\Users\ZILE\Desktop\IKP\IKPProjekat\Projekat\Debug\Client.exe
Odaberite ulogu:
    1 -> vodja
    2 -> igrac
-> 1
Uspesno registrovani kao vodja!
Unesite zamisljen broj: 15
Unesite opseg brojeva!
Od: 10
Do: 100
Poruka od igraca -1 je -858993460
Poruka od igraca 1 je 15
Poruka od igraca 2 je 19
Od igraca id = 1 primljen broj 15 i on je TACAN
Od igraca id = 2 primljen broj 19 i on je VECI
Pritisni dugme za gasenje programa !
```

Slika 14

## Strukture podataka:

Klijent veza(client\_veza)-struktura sadži ID(jedinstven identifikator

```
typedef struct client_veza_st {
    int id;
    SOCKET socket;
    HANDLE thread;
    client_veza_st* next;
}client_veza;
```

Slika 15

klijenta), SOCKET(konekcija sa klijentom), HANDLE(thread namenjen za prijem poruke). Čuva veze sa klijentom i služi nam da napravimo listu konektovanih igrača kako bi server imao informacije gde treba da prosledi poruke.

Opseg(opseg)-struktura sadrži dva INT-a koja nam označavaju u kom opsegu se nalazi broj za pogađanje.

```
typedef struct opseg_st {  
    int od;  
    int doo;  
}opseg;
```

Slanje ove strukture označava početak igre.

Slika 16

Poruka igrača(poruka\_igraca)-struktura sadrži ID(jedinstven

```
typedef struct poruka_igraca_st {  
    int id;  
    int broj;  
    poruka_igraca_st* next;  
}poruka_igraca;
```

identifikator klijenta), INT koji predstavlja broj koji je klijent pogađao. Ovu strukturu server smesta u red pogađanja i salje vodi koji je proverava.

Slika 17

Odgovor vođe(odgovor\_vodje)-struktura sadrži ID(jedinstven

```
typedef struct odgovor_vodje_st {  
    int id;  
    odgovor odg;  
    odgovor_vodje_st* next;  
}odgovor_vodje;
```

identifikator klijenta), ENUM koji predstavlja odgovor vođe(VEĆE, MANJE, TAČNO, IZGUBIO). Pakujemo je u red odgovora.

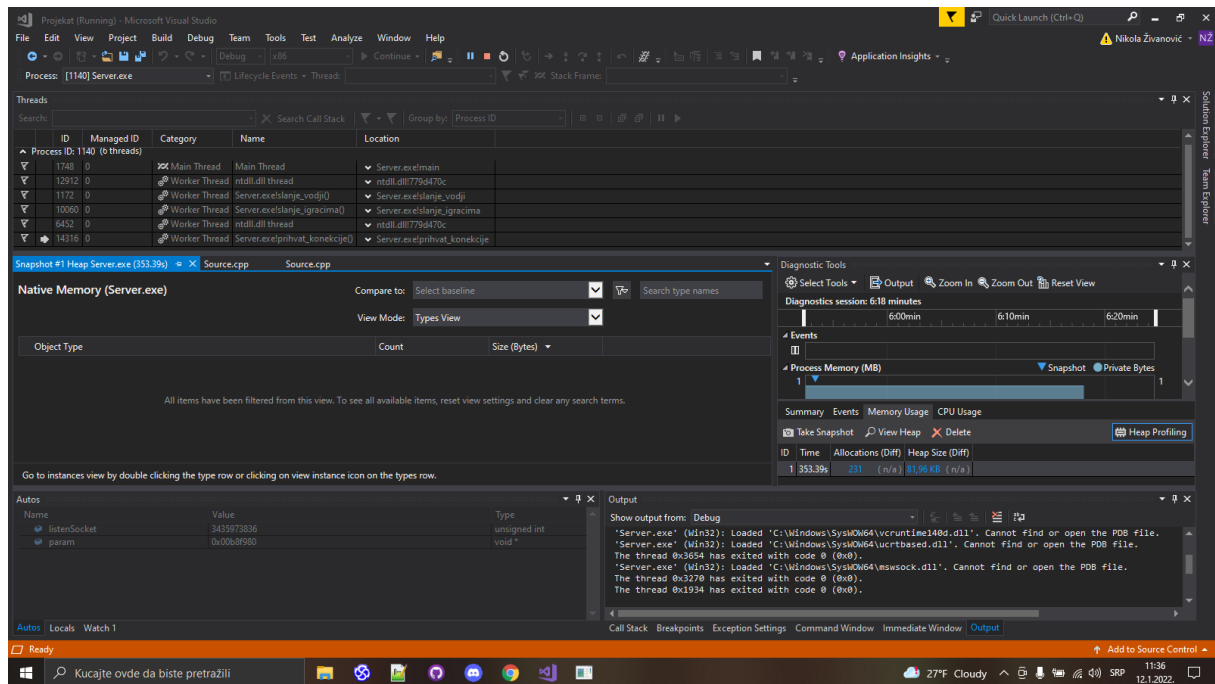
Slika 18

```
enum odgovor { VECE = 1, MANJE, TACNO, IZGUBIO};
```

Slika 19

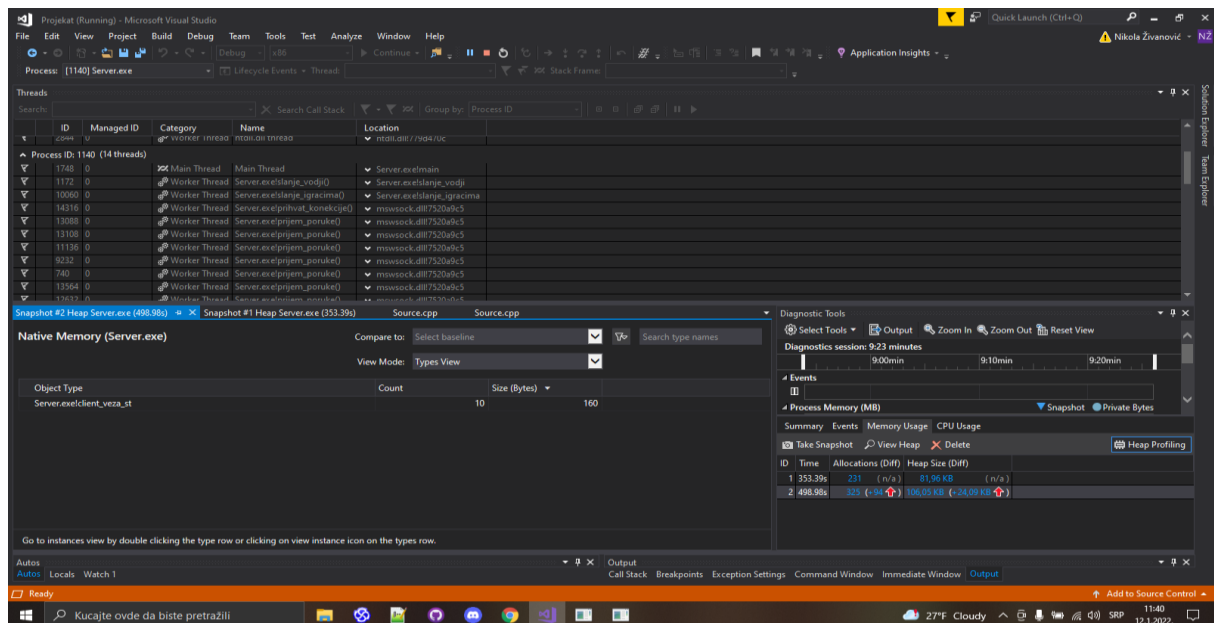
## Rezultati testiranja:

Testovi su realizovani tako što su pokrenuti klijenti(10 instanci) i jedan server i izabran jako veliki opseg(2000000000), nakon čega se krenulo u igru. Svi klijenti su koristili algoritam za pogađanje broja.



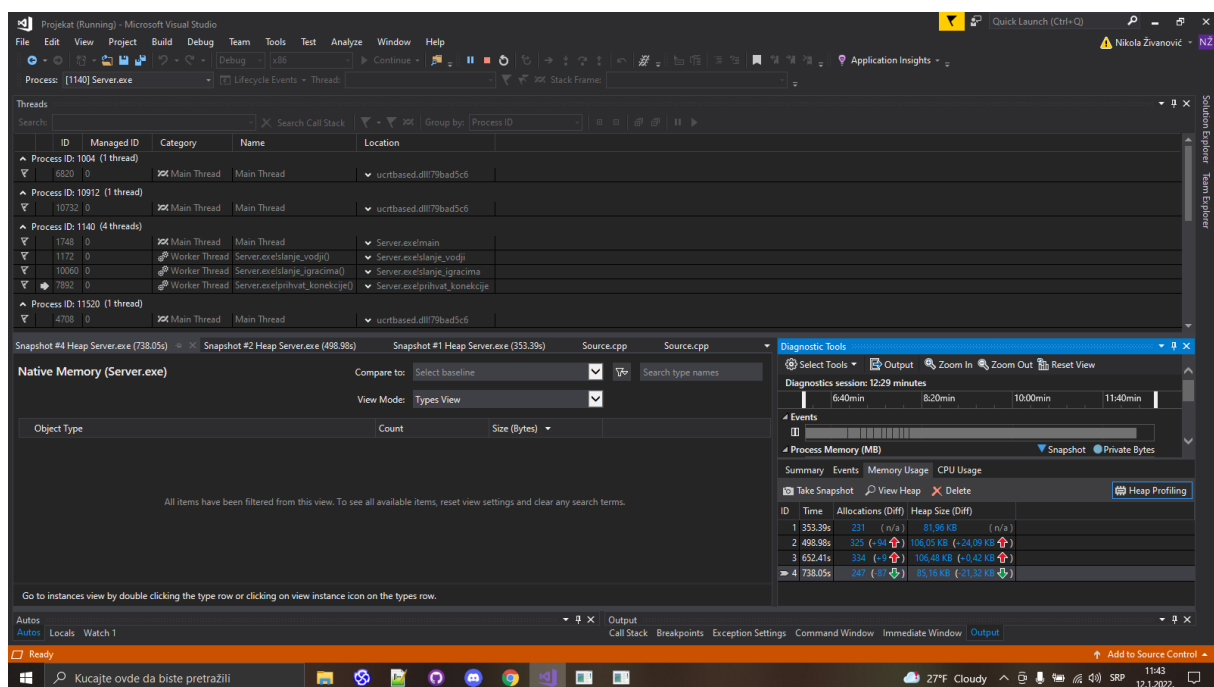
Slika 20

Kao što je predviđeno dizajnom prilikom pokretanja aktiviraju se pored Main thread-a još dodatna tri thread, jedan za prihvati konekcije, drugi za slanje poruka vodi i treći za slanje poruka igračima. Posto klijenti još uvek nisu konektovani nemamo strukture ni dodatna zauzeća memorije.



Slika 21

Prilikom konektovanja svih klijenata vidimo da su aktivirani novi thread-ovi za svakog klijenta i da je zauzeta određena memorija.



Slika 22

Kada se program završi vidimo da su svi thread-ovi klijenata pogašeni da su sve strukture oslobođene kao i dodatna memorija. Vidimo da postoji curenje memorije od nekoliko KB-a.

## Zaključak:

Na osnovu testova zaključujemo da igra funkcioniše kako treba ali se vidi malo curenje memorije koje može biti proizvedeno od strane dodatno zauzetih biblioteka ili nekih promenljivih koje su korišćene tokom igre. Algoritam je efikasan za jako velike opsege.

Jedan od mogućih načina za unapređenje jeste taj da se uvede vremensko ograničenje za klijenta, odnosno koji vremenski interval mu se daje za slanje odgovora, kao i moguće ograničenje na serveru tj vreme čekanja na sve odgovore kako ne bi došlo do situacije da jedan klijent blokira celu igru.

Drugi način unapređenja jeste da klijenti nezavisno šalju poruke i dobijaju odgovore i time bi se ponovo izbegla mogućnost da jedan klijent blokira igru.