

**»Mladi za napredek Maribora 2021«
38. srečanje**

RULETA

Raziskovalno področje: Elektrotehnika, elektronika

Raziskovalna naloga

PROSTOR ZA NALEPKO

Avtor: ALJAŽ REMS
Mentor: BOJAN DEŽMAN
Šola: SREDNJA ELEKTRO-RA UNALNIŠKA ŠOLA MARIBOR
Število to k: 125 / 170

Maribor, 2021

KAZALO VSEBINE

1. POVZETEK	4
2. ZAHVALA	4
3. UVOD	5
4. UČENJE STANDARDOV	6
5. IZDELAVA PROGRAMA	7
5.1 Načrtovanje	7
5.2 Programiranje	7
5.2.1 Funkcija Main()	7
5.2.2 Funkcija Game()	7
5.2.3 Funkcija RAND_NUMBER()	10
5.2.4 Funkcija RED_BLACK()	10
5.2.5 Funkcija STAVA_NUMBER STAVA_NAPACNA	11
5.3 Testiranje	11
ZAKLUČEK	12
VIRI	13

KAZALO SLIK

Slika 1: Primer ruletine mize (google images)	5
Slika 2: Dva primera učenja metod (lasten vir)	6
Slika 3: Funkcija main (lasten vir)	7
Slika 4: If stavek za določeno število (lasten vir)	7
Slika 5: If stavek za parno/neparno število (lasten vir)	8
Slika 6: If stavek za barvno stavo (lasten vir)	9
Slika 7: Funkcija RAND_NUMBER (lasten vir)	10
Slika 8: Funkcija IS_RED (lasten vir)	10
Slika 9: Funkcija STAVA_NUMBER in STAVA_NAPACNA (lasten vir)	11
Slika 10: Testiranje programa (lasten vir)	11

1. POVZETEK

Za projekt bom uporabil preposto igro Ruleta z pomočjo katere se bom naučil standarde programiranja in boljše načine reševanja problemov. Menim, da se veliko ljudi loti programiranja zelo ne organizirano in se mi zdi da je to tudi veliki problem. Zato bom sam tudi stestiral ali je res zelo pomembna organizacija, popolno razumevanje vseh stvari v programiranju tako zelo pomembno .

2. ZAHVALA

Najprej se bi rad zhvalil mentorju pri organiziranju, pomoči, podpori in spodbujandi pri tej nalogi. Velika zahvala tudi šoli katera je vložila trud in me naučila osnove programiranja in mi predala trenutno znanje.

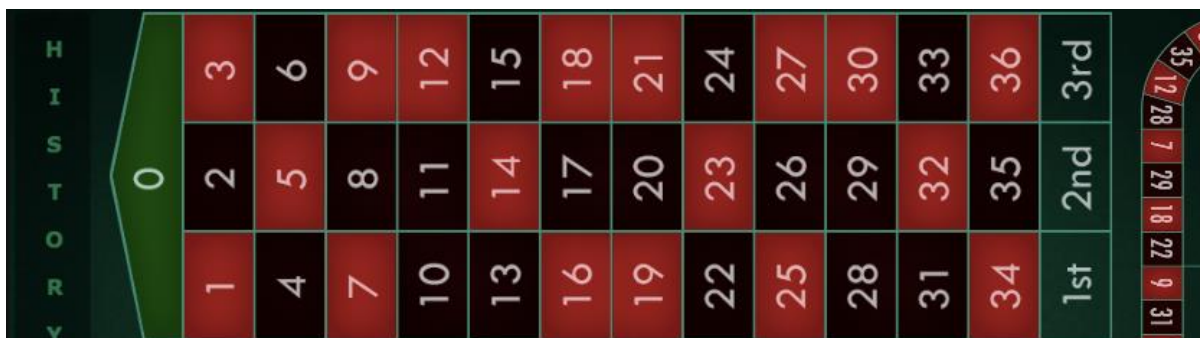
3. UVOD

Igra ruleta je zasnovana po originalni igri, katera se igra v kazinojih, spletnih straneh, klubih itd... Osnovna pravila rulete bom tudi uporabil v moji različici igre, katero bom spisal s pomočjo programskega jezika C++. Torej bistvo igre je, da bo lahko igralec stavil denar na tri različne načine:

- **Glede na barvo**
- **Na točno število**
- **Parno ali ne parno**

Če igralec pravilno stavi dobi za določeno število povišano nagrado. Po končani rundi se lahko igralec odloči ali še bo igral igro še en krat ali ne. Če igralec doseže 0\$ se igra avtomatsko zakluči.

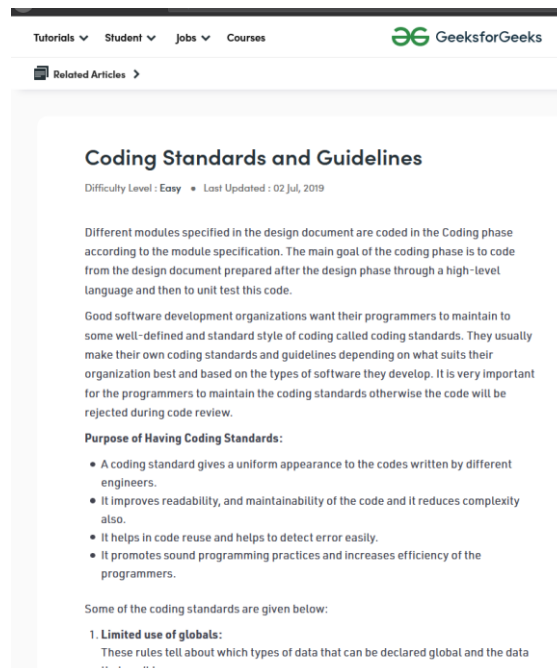
V tem projektu se bom probal izboljšati na vseh področjih programiranja torej probal bom pridobiti čim boljšo organizacijo dela, izglednost kode, optimalnost kode, dokumentacijo itd. V tem projektu bom ugotovil ali to zelo vpliva na moj »Work flow« in kakšni so plusi in kakšni minusi tekega načina dela. Predvidevam tudi, da bo mi to v korist tudi v prihodnosti vsaj vsako znanje katerega sem do zdaj pridobil mi je prišlo v korist in mi omogočalo hitrejše programiranje, hitrejše razmišljanje (reševanje problemov in boljši načini kako se soočati z njimi) ter hitrejše odpravljanje napak.



Slika 1: Primer ruletine mize (google images)

4.UČENJE STANDARDOV

Preden sem začel z mojim projektom sem si ogledal nekaj spletnih strani in video posnetkov glede izboljšave na področju programiranja. Saj programer, ki se ukvarja več let z tem ima dosti več izkušen od mene in mi lahko, da dosti več nasvetov za programiranje. Na internetu imamo dostop do vseh informacij katere potrebujemo za izboljšavo na področju porgramiranja zato se lahko bilo kdo izboljša ali nauči vse o tem.



The screenshot shows a webpage from GeeksforGeeks. The header includes navigation links for Tutorials, Student, Jobs, and Courses, along with the GeeksforGeeks logo. Below the header, there's a section for 'Related Articles' and a main article titled 'Coding Standards and Guidelines'. The article is marked as 'Difficulty Level : Easy' and 'Last Updated : 02 Jul, 2019'. The content discusses the importance of coding standards in software development, mentioning that they help in maintaining a uniform appearance, improving readability, and reducing complexity. It also lists the purpose of having coding standards and provides a list of standards, starting with '1. Limited use of globals:'. The article is written by Brian Voong and is copyrighted by 'Lets Build That App'.

Coding Standards and Guidelines

Difficulty Level : Easy • Last Updated : 02 Jul, 2019

Different modules specified in the design document are coded in the Coding phase according to the module specification. The main goal of the coding phase is to code from the design document prepared after the design phase through a high-level language and then to unit test this code.

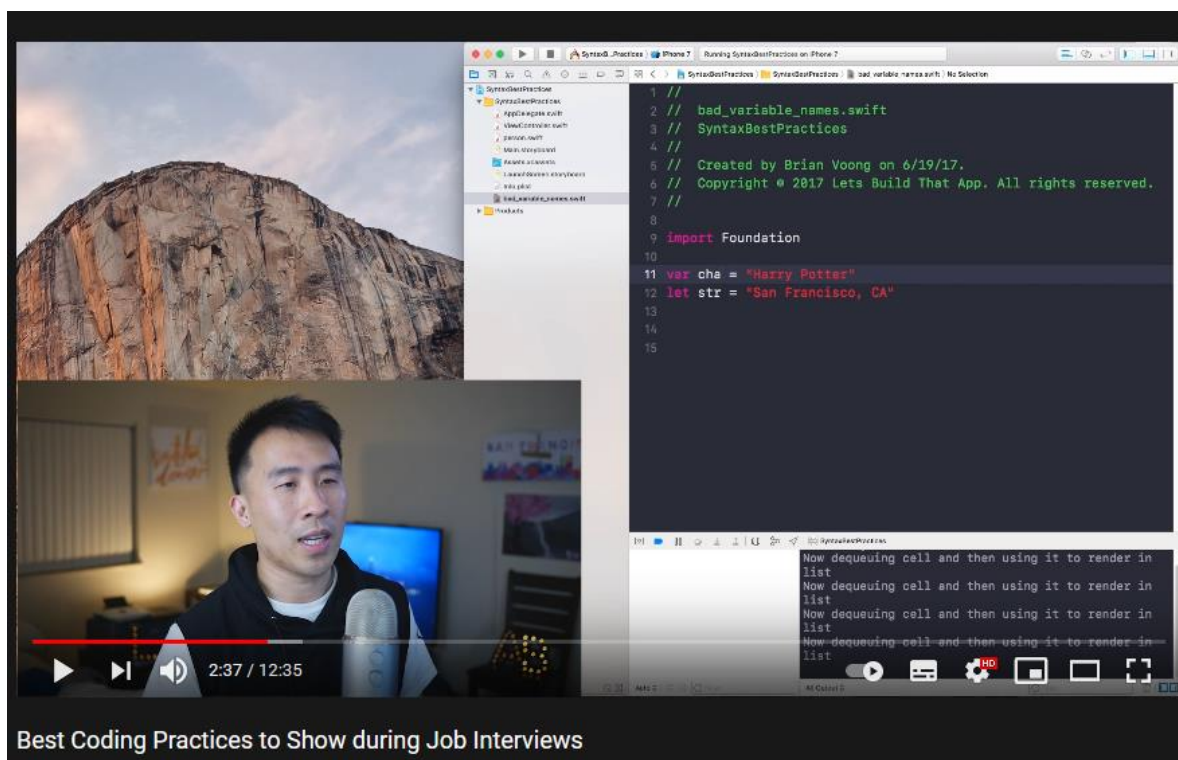
Good software development organizations want their programmers to maintain to some well-defined and standard style of coding called coding standards. They usually make their own coding standards and guidelines depending on what suits their organization best and based on the types of software they develop. It is very important for the programmers to maintain the coding standards otherwise the code will be rejected during code review.

Purpose of Having Coding Standards:

- A coding standard gives a uniform appearance to the codes written by different engineers.
- It improves readability, and maintainability of the code and it reduces complexity also.
- It helps in code reuse and helps to detect error easily.
- It promotes sound programming practices and increases efficiency of the programmers.

Some of the coding standards are given below:

- 1. Limited use of globals:**
These rules tell about which types of data that can be declared global and the data that can't be.



Slika 2: Dva primera učenja metod (lasten vir)

5. IZDELAVA PROGRAMA

5.1 Načrtovanje

Prva stvar z katero sem začel je načrtovanje odločil sem se kaj vse bo igra zajemala, katere opcije bo vse imel igralec in na kakšen način se bi sploh lotil naloge.

Odločil sem se, da glavna valuta v igri bo denar. Objektiv igralaca bo, da zbere čim več denarja. Vendar če pristane na 0 \$ zgubi igro.

Nato sem si pogledal na katere načine lahko vse stavljaš v ruleti in katera števila so vse na voljo. Torej zabeležil sem si da lahko igralec stavlja na: parna in neparna števila, na barve in na določeno število. Glede na to katero možnost si izbere za toliko krat se bo mu njegov dobiček povečal. Nato sem si še pogledal od katerega do katerega števila lahko stavljaš v ruleti (0-36) in sem mel načrt moje igre končan.

5.2 Programiranje

5.2.1 Funkcija Main()

Začel sem s preprosto funkcijo main, katera te izpiše ali igralec hoče igrati ruleto ali ne. In glede na njegovo odločitev pokliče funkcijo, da začne igro ali pa zakliče samo program. V main funkciji sem tudi nastavo, da igralec značne 2000\$.

```
int main()
{
    char odlocitev;
    cout << "Ali Zelite igrati ruleto" << endl;
    cout << "Da [D] / NE [N]: " ;
    cin >> odlocitev;
    if(toupper(odlocitev) == 68){
        Game(2000);
    }
}
```

Slika 3: Funkcija main (lasten vir)

5.2.2 Funkcija Game()

Je glavna funkcija, najprej sem definiral, da funkcija protebuje int, ko jo pokličeš v tem primeru je to uporabnikovo denarno stanje. Igralec vpiše njegovo stavo in če ta preseže njegov limit ga opozori in funkcijo zažene znova. Ko igralec uspešno stavi mu izpiše navodila kako izbere tip igre.

```
cout << "Vnesi [K] ce zelis staviti na parno/neparno stevilo ali [S] na doloceno stevilo stave na barvo [B]: ";
```

Ko igralec izbere tip igre se naključno generira število z pomočjo funkcije RAND_NUMBER() katero sem spisal in bom kaneje natančneje obrazložil.

Glede na uporabnikovo izbiro igre sem z pomočjo if stavkov preveril ali je uporabni zadel stavo ali ni. Pri stavi na število sem preveril ali je stavljeno število enako naključno generiranemu in če je se dobiček za sedem krat poveča drugače pa igralec izgubi njegov vližen denar.

```
if(Player_Number_Bet != Random_Number)
{
    cout << "Zgubili ste " << Player_money_bet << "$" << endl;
    Player_Money -= Player_money_bet;
}
else
{
    cout << "Zmagali ste " << 7*Player_money_bet << "$" << endl;
    Player_Money += 7*Player_money_bet;
}
```

Slika 4: If stavek za določeno število (lasten vir)

Če je igralec stavil na parno ali ne parno sem delil z pomočjo % in dobil ostanek in če je ostanek večji od nič pomeni, da je število neparno in na osnovi tega sem spisal kodo

```
if(toupper(Game_Type) == 75)
{
    cout << "Ali bi stavili na parna stevila [P] ali neparna [N]? ";
    cin >> ParnoNeParno;
    cout << "Zoga se je ustavila na: " << Random_Number << endl;
    if(toupper(ParnoNeParno) == 80)
    {
        if(Random_Number%2 == 0)
        {
            cout << "Zmagali ste: " << Player_money_bet << "$"<< endl;
            Player_Money += Player_money_bet;
        }
        else
        {
            cout << "Zgubili ste: " << Player_money_bet << "$"<< endl;
            Player_Money -= Player_money_bet;
        }
    }
    if(toupper(ParnoNeParno) == 78)
    {
        if(Random_Number%2 == 0)
        {
            cout << "Zgubili ste: " << Player_money_bet << "$"<< endl;
            Player_Money -= Player_money_bet;
        }
        else
        {
            cout << "Player_Moneyli ste: " << Player_money_bet << "$"<< endl;
            Player_Money += Player_money_bet;
        }
    }
}
}
```

Slika 5: If stavek za parno/neparno število (lasten vir)

Če igralec stavi na barvo sem spisal eno preprosto funkcijo, ki preveri na kateri barvi se je krogla usavla in vrne true ali false glede na to.

```
if(toupper(Game_Type) == 66)
{
    cout << "Ali bi stavili na crno polje [C] ali rdeco [R] ali zelena [G]? ";
    cin >> ParnoNeParno;
    cout << "Zoga se je ustavila na: " << Random_Number << endl;
    if(toupper(ParnoNeParno) == 71)
    {
        if(Random_Number == 0)
        {
            cout << "Player_Moneyli ste: " << Player_money_bet*30 << "$"<< endl;
            Player_Money += Player_money_bet*30;
        }
        else
        {
            cout << "Zgubili ste: " << Player_money_bet << "$"<< endl;
            Player_Money -= Player_money_bet;
        }
    }

    if(toupper(ParnoNeParno) == 82)
    {
        if(RED_BLACK(Random_Number) == true)
        {
            cout << "Player_Moneyli ste: " << Player_money_bet << "$"<< endl;
            Player_Money += Player_money_bet;
        }
        else
        {
            cout << "Zgubili ste: " << Player_money_bet << "$"<< endl;
            Player_Money -= Player_money_bet;
        }
    }

    if(toupper(ParnoNeParno) == 67)
    {
        if(RED_BLACK(Random_Number) == true)
        {
            cout << "Zgubili ste: " << Player_money_bet << "$"<< endl;
            Player_Money -= Player_money_bet;
        }
        else
        {
            cout << "Player_Moneyli ste: " << Player_money_bet << "$"<< endl;
            Player_Money += Player_money_bet;
        }
    }
}
```

Slika 6: If stavek za barvno stavo (lasten vir)

5.2.3 Funkcija RAND_NUMBER()

Funkcijo random number sem spisal zato, da se mi koda par krat ne ponovi ampak samo pokličem funkcijo ko rabim random number med 0 – 36

```
int RAND_NUMBER()
{
    int const MIN_ST = 0, MAX_ST = 36;
    srand(time(NULL));
    int Random_Number = rand() % (MAX_ST - MIN_ST + 1) + MIN_ST;

    return Random_Number;
}
```

Slika 7: Funkcija RAND_NUMBER (lasten vir)

5.2.4 Funkcija RED_BLACK()

Funkcijo RED_BLACK sem naredil z razlogom, da nerabim tlačit vsega v eno funkcijo in, da je koda bolj pregleda. Deluje na osnovi tega, da sem ustvaril array v katerega sem dal vsa števila, ki so rdeča in če je rdeče število vrne true. V primeru, da je črno vrne false.

```
bool RED_BLACK(int random_number)
{
    int red[18] = {1,3,5,7,9,12,14,16,18,21,23,25,27,28,30,32,34,36};
    bool check = false;
    if(random_number != 0)
    {
        for(int i = 1;i<=18;i++)
        {
            if(red[i] == random_number)
            {
                check = true;
            }
        }
    }
    return check;
}
```

Slika 8: Funkcija IS_RED (lasten vir)

5.2.5 Funkcija STAVA_NUMBER || STAVA_NAPACNA

Funkcija STAVA_NUMBER je poklicana kadar igralec hoče staviti na število in v primeru, da ne vpiše števila med 0 – 36 pokliče funkcijo STAVA_NAPACNA in mu napiše da mora staviti med 0 in 36 če tega ne naredi bo se funkcija ponavljala.

```
int STAVA_NAPACNA()
{
    int Player_Number_Bet;

    cout << "OBVEZNO IZBERITE STEVILO MED 0 in 36: ";
    cin >> Player_Number_Bet;
    if( Player_Number_Bet > 36 || Player_Number_Bet < 0 )
    {
        STAVA_NAPACNA();
    }
    return Player_Number_Bet;
}

int STAVA_NUMBER(){
    int Player_Number_Bet;
    cout << "Na katero stevilo zelite staviti: ";
    cin >> Player_Number_Bet;

    if( Player_Number_Bet > 36 || Player_Number_Bet < 0 ){
        STAVA_NAPACNA();
    }
    return Player_Number_Bet;
}
```

Slika 9: Funkcija STAVA_NUMBER in STAVA_NAPACNA (lasten vir)

5.3 Testiranje

Med programiranjem sem naredil nekaj malih napak vendar sem jih med probo igre odpravil (Bug fix). Torej igro sem tudi sam sprobalo kako deluje in tukaj sem še priložil nekaj slik.

```
Ali Zelite igrati ruleto
Da [D] / NE [N]: D
-----[To je Roulette]-----
Vpisite stavo:200
Vnesi [K] ce zelis staviti na parno/neparno stevilo ali [S] na doloceno stevilo stave na barvo [B]: S
Na katero stevilo zelite staviti: 36
Zoga se je ustavila na : 14
Zgubili ste 200$
Tvoje denarno stanje je: 1800 $
Ali Zelite igrati ruleto se En Krat
Da [D] / NE [N]:

Vpisite stavo:2000
Presezek limita ! tvoj limit je: 1800
-----[To je Roulette]-----
Vnesi [K] ce zelis staviti na parno/neparno stevilo ali [S] na doloceno stevilo stave na barvo [B]: B
Ali bi stavili na crno polje [C] ali rdeco [R] ali zelena [G]? R
Zoga se je ustavila na: 4
Zmagali ste: 200$
Tvoje denarno stanje je: 2200 $
Ali Zelite igrati ruleto se En Krat
Da [D] / NE [N]:
```

Slika 10: Testiranje programa (lasten vir)

ZAKLUČEK

Skozi to nalogo sem se naučil dosti načinov kako izboljšati moj »work flow« pri programiranju. Predvsem bi rekel, da mi je pomagalo pri organizacija kar sem že predvideval preden sem začel z to nalogo vendar sem ugotovil, da z organizacijo pride tudi boljša optimalnost kode in bolši načini reševanja problemov. Z tem ko sem kodo pisal po nekakšnih standardih sem se hitreje znašel v njej hitreje lahko odpravljal težave in s tem tudi prihranil nekaj časa. Preden sem začel projekt sem mislil, da bom z tem pristopom do programiranja porabil dosti več časa, saj sem mislil, da dokumentacija dela zavzame dosti več časa, vendar na koncu sem ugotovil, da sem pristal na enakem. Vendar pa so z takim načinom dela prišli vlki plusi:

- Preglednost kode
- Optimalnost kode
- Hitrejše debuganje
- Hitrejše razmišljanje
- Neponavljajoča koda
- Razumevanje funkcij

Razumevanje funkcij je bil eden največjih napredkov saj sem ugotovil, da koda postane toliko krat bolj pregledna in če se funkcija več krat ponovi jo lahko samo pokličem. S tem sem si prihranil veliko časa in sem porabil veliko manj vrstic.

Iz moje izkušnje, predlagam vsem, da se že od začetka programiranja začnejo učiti organizacijo kode in se naučijo standardov kateri so pri tem postavljeni, da si pogledajo kako se bolj izkušeni programerji lotijo dela. To je možno preko youtube, v šoli ali pa z pregledovanjem njihove kode. Saj menim, da bi vsak prihranil veliko časa in živcov.

Torej na koncu koncev sem ugotovil, da se je splačalo pregledati par spletnih strani in videjev glede programiranja. Saj sem pridobil kar napredno znanje iz tega področja.

VIRI

<https://www.youtube.com/watch?v=gnZh0CqOCA0> (Dostop 1.3.2021)

<https://www.youtube.com/watch?v=Q7mWQ6Uf0uw> (Dostop 1.3.2021)

<https://www.youtube.com/watch?v=kOfzwgnvp6A> (Dostop 1.3.2021)

<https://www.geeksforgeeks.org/coding-standards-and-guidelines/> (Dostop 1.3.2021)