Sveučilište Josipa Jurja Strossmayera u Osijeku

Elektrotehnički fakultet

Seminar iz kolegija „Algoritmi i strukture podataka“

Skakačev obilazak

Knight's tour

Izradio: Mario Benčić, bacc.ing.el.

Osijek, 2011.

SADRŽAJ

UVOD

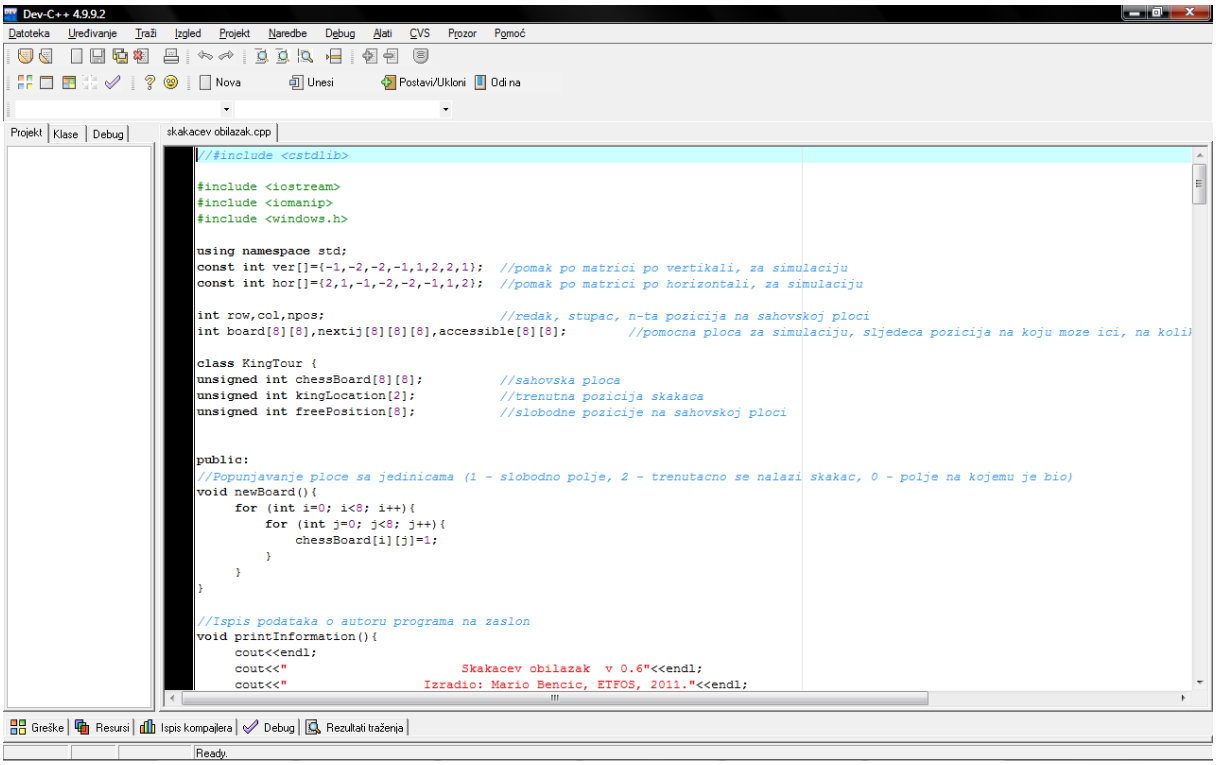
Skakačev obilazak je matematički problem koji uključuje skakača i šahovsku ploču. Skakač se postavi na praznu šahovsku ploču i kreće se prema pravilima šahovske igre. Problem skakačevog obilaska sastoji se u nalaženju niza poteza skakača tako da sva polja šahovske ploče budu posjećena točno jednom. Program treba pokazati da je skakač u mogućnosti obići cijelu ploču u 64 poteza, tj. da svako polje bude posjećeno samo jednom. Skakač se kreće u obliku slova 'L' te obilazak počinje na zadanoj lokaciji. Redosljed obilazaka polja je animiran.

DEV C++

Programsko okruženje koje je korišteno za pisanje programa naziva se DEV C++. Program je besplatan, ugodnog izgleda, na hrvatskom jeziku i malih dimenzija.

Iako je taj program relativno malih dimenzija, posjeduje sve osobine koje su potrebne za učenje programiranja, a može poslužiti i za manje složeno profesionalno programiranje.

Slika 1.1. Programski jezik DEV C++



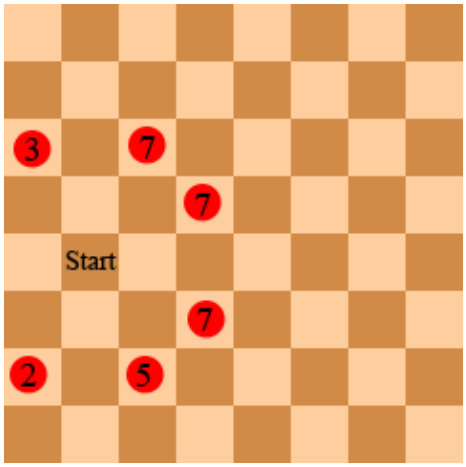
Slika 1.2. Sučelje programa DEV C++

WARNSDORFF-ov ALGORITAM

Postoji više algoritama za rješavanje ovog matematičkog problema. Jedan od njih je i brute-force metoda koja ispituje sve moguće kombinacije poteza da bi cijela ploča bila obiđena.

Važno je napomenuti da se ova metoda može koristiti samo za manje ploče jer je memorijski veoma zahtjevna. Za uobičajenu 8x8 ploču, postojalo bi otprilike $4 \cdot 10^{51}$ kombinacija za provjeru, tako da se taj algoritam inače ne koristi.

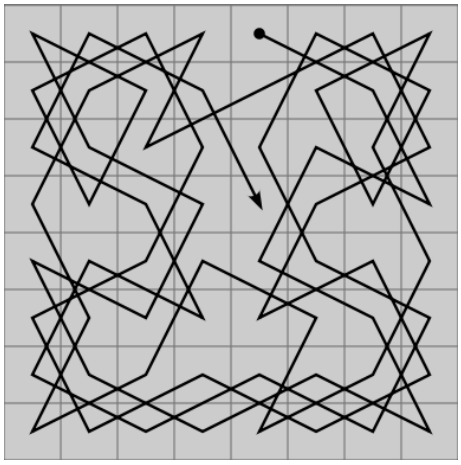
Warnsdorff-ov algoritam je metoda za rješavanje problema skakačevog obilaska bazirana na pravilu provjere polja odmah dostupnih skakaču s kojih bi mogao nastaviti kretanje, ali sa što manjim brojem polja na koja bi mogao stati, nakon što stane na to polje.



Slika 2.1. Metoda Warnsdorff-ovog algoritma

Algoritam će napraviti takvu provjeru za bilo koju početnu poziciju na ploči. Pronalaze se sva polja koja mogu biti posjećena u jednom potezu i broj poteza koje bi skakač mogao obaviti s tog polja. Zatim algoritam premješta skakača na polje s najmanjim brojem sljedećih poteza. Proces se ponavlja sve dok sva polja ne budu posjećena.

Inače, na ploči 8x8 postoji točno 26 534 728 821 064 kombinacija obilazaka ploče (dva obilaska koja su prošla istim putem u suprotnom smjeru se broje posebno, kao što su rotacija i refleksija), ali se ovim algoritmom taj broj uvelike smanjuje.



Slika 2.2. Primjer jednog obilaska šahovske ploče u 64 poteza

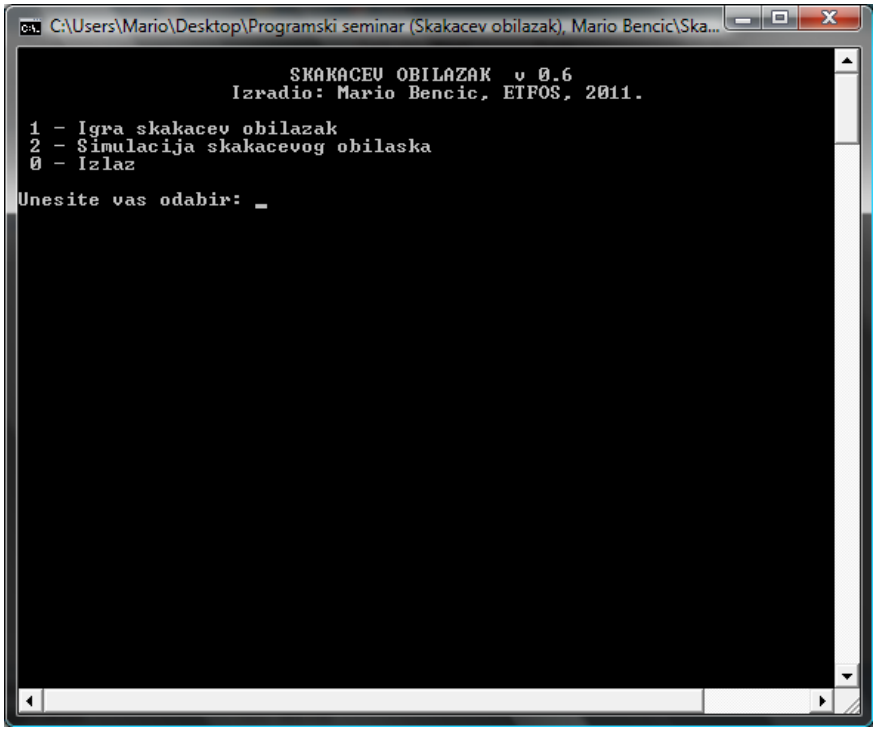
Neke definicije:

- Položaj Q je dostupan skakaču na poziciji P ako se skakač može prebaciti na Q u jednom potezu i ako Q već nije bio posjećen.
- Dostupnost položaja P je broj polja dostupnih tom polju.

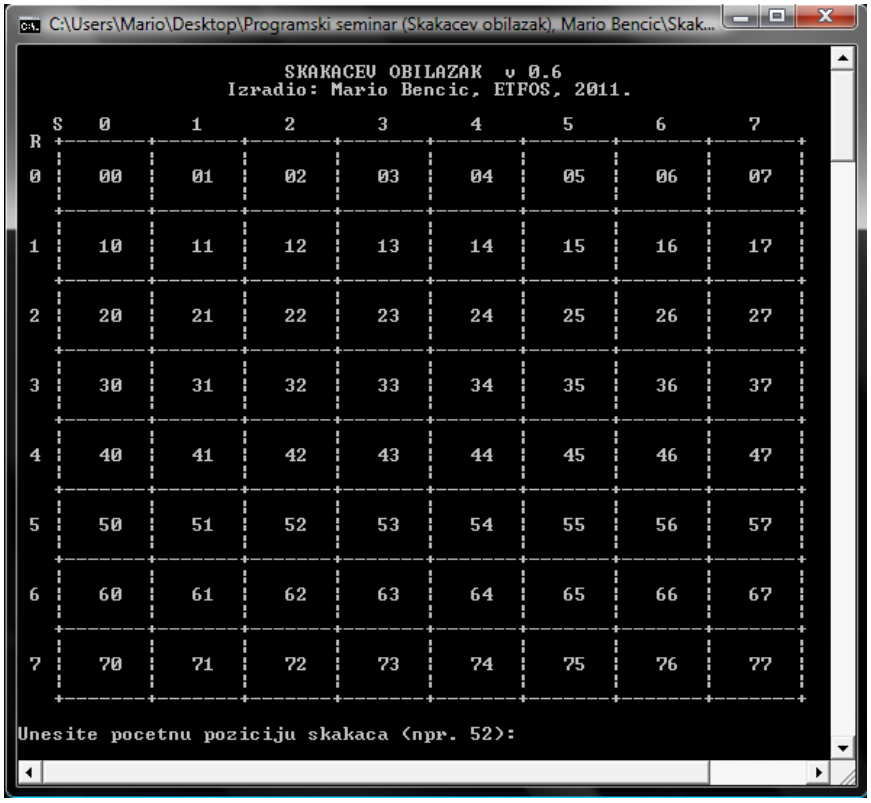
Algoritam:

1. Postavi P kao proizvoljnu poziciju skakača na ploči
2. Označi na ploči poziciju P s brojem „1“
3. Za svaki potez od broja 2 pa do ukupnog broja polja na ploči:
    a) neka S bude broj pozicija dostupnih sa početne pozicije
    b) postavi P u poziciju S sa najmanjom pristupačnošću
    c) označi na ploči položaj P s rednim brojem poteza
4. Vrati označenu ploču – svako polje označi rednim brojem poteza u kojem je posjećeno

PROGRAM: SKAKAČEV OBILAZAK



Slika 3.1. Izbornik programa „Skakačev obilazak“



Slika 3.2. Igra skakačevog obilaska

Slika 3.2. prikazuje igru skakačevog obilaska gdje se može ručno probati obići ploča upisivanjem brojeva polja. Ukoliko se ploča uspješno obiđe u 64 poteza, na ekranu se ispisuje:

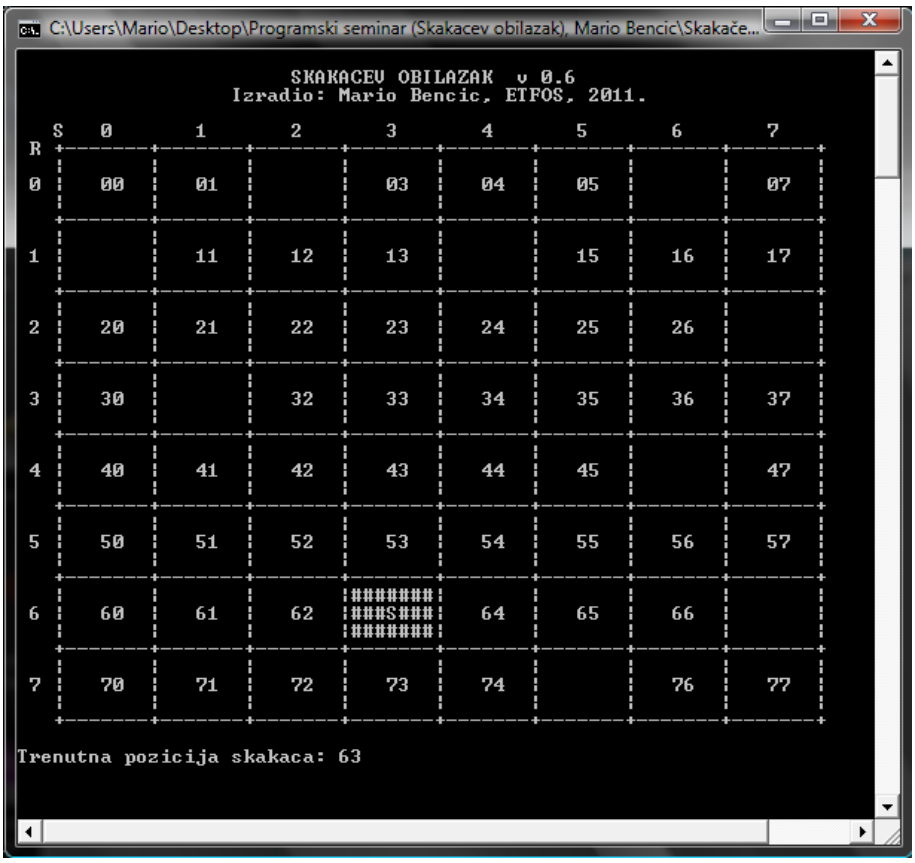
BRAVO!!! USPJESNO STE OBISLI PLOCU!!! CESTITAMO!!!

Ukoliko se ploča ne uspije obići uz zadani uvijet na ekranu će se ispisati:

KRAJ IGRE!!!!

Nazalost niste uspjeli doci do kraja.

Odmorite skakaca pa pokusajte ponovno!



Slika 3.3. Simulacija skakačevog obilaska

Slika 3.3. prikazuje simulaciju skakačevog obilaska koja se poziva biranjem broja 2 u glavnom izborniku. Nakon toga je potrebno upisati proizvoljno broj polja s kojeg skakač kreće te simulacija započinje. Trenutna pozicija skakača je označena slovom „S“, a polja na kojima je skakač bio su prazna. Po završetku simulacije ispisuju se redom sva 64 polja na kojima je skakač bio u obilasku cijele ploče.

ZAKLJUČAK

Skakačev obilazak je prije svega matematički problem čiji se pokušaji rješavanja bilježe još od devetog stoljeća. Do danas su razvijeni brojni algoritmi koji taj problem rješavaju u kratkom vremenu.

Ovim programskim seminarom je pokazano da je šahovska figura „Skakač“ doista u mogućnosti da obiđe cijelu ploču, odnosno 64 polja, u 64 poteza. To je postignuto korištenjem Warnsdorff-ovog algoritma koji je jedan od najjednostavnijih i najbržih. Njegov princip rada zasniva se na provjeri svih polja koja su dostupna skakaču u jednom potezu, te odabir onog s najmanjom dostupnošću. Tako se postupno eliminiraju najteže dostupna polja te ostaju samo lakše dostupna polja. Ovim postupkom je moguće ostvariti ukupno 64 različita obilaska šahovske ploče.

PRILOZI

Programski kod skakačevog obilaska:

```
//#include <cstdlib>
#include <iostream>
#include <iomanip>
#include <windows.h>
using namespace std;
const int ver[]={-1,-2,-2,-1,1,2,2,1};  //pomak po matrici po vertikali, za simulaciju
const int hor[]={2,1,-1,-2,-2,-1,1,2};  //pomak po matrici po horizontali, za simulaciju
int row,col,npos;                   //redak, stupac, n-ta pozicija na sahovskoj ploci
int board[8][8],nextij[8][8][8],accessible[8][8];       //pomocna ploca za simulaciju, sljedeca pozicija na koju moze ici, na koliko polja se sa toga polja mozemo pomaknuti
class KnightTour {
unsigned int chessBoard[8][8];       //sahovska ploca
unsigned int knightLocation[2];       //trenutna pozicija skakaca
unsigned int freePosition[8];        //slobodne pozicije na sahovskoj ploci
public:
//Popunjavanje ploce sa jedinicama (1 - slobodno polje, 2 - trenutacno se nalazi skakac, 0 - polje na kojemu je bio)
void newBoard(){
   for (int i=0; i<8; i++){
      for (int j=0; j<8; j++){
         chessBoard[i][j]=1;
      }
   }
}
//Ispis podataka o autoru programa na zaslon
void printInformation(){
   cout<<endl;
   cout<<"                  SKAKACEV OBILAZAK  v 0.6"<<endl;
   cout<<"              Izradio: Mario Bencic, ETFOS, 2011."<<endl;
}
/*Ispis ploce na zaslon. Na polja na koja moze stati skakac
upisuju se adrese, polja na kojima je bio ostaju prazna,
polje na kojemu se nalazi ima oznaku S
*/
void printBoard(){
      string num = "   S  0     1     2     3     4     5     6     7";
       string hor = " +-------+-------+-------+-------+-------+-------+-------+-------+";

       cout<<endl;
       cout<<num<<endl;
       cout<<" R"<<hor<<endl;
```

```
        for (int z=0; z<8;z++){   //Petlja za pomak po redovima

          for (int j = 0; j < 3; j++){ // Petlja za ispis redaka kocke
             if (j != 1) {
               cout<<"   ";
               } else {
                cout<<" "<<z<<" ";   // U srednjem redu kocke ispiši redni broj kocke (redak)
                }
         for (int i=0; i<9;i++){ //Petlja za pomak po stupcima

           if(chessBoard[z][i] == 1 && j == 1 && i <= 7){  //Polje na koje moze doci skakac, ispisuje se njegova numericka lokacija
           cout<<"|   "<<z<<i<<"  ";
           } else if (chessBoard[z][i] == 0 && j == 1 && i <= 7){  //Polje na kojemu je skakac bio, ostaje prazno
                cout<<"|      ";
           } else if    (chessBoard[z][i] == 2 && i <= 7 && j == 1){
                cout<<"|###S###";
           }else if (chessBoard[z][i] == 2 && i <= 7) {  //Polje na kojemu se trenutačno nalazi skakac, ima oznaku S
                cout<<"|#######";
           } else {
            cout<<"|      ";
           }
           }
           cout<<endl;
           }
         cout<<"  "<<hor<<endl;
        }
}
//Pomak skakaca po ploci, i pohrana trenutne lokacije u vektor - knightLocation
void shiftKnight(unsigned int rc){
    unsigned int r = rc/10;   // izdvajanje reda
    unsigned int c = rc%10;   // izdvajanje stupca
    chessBoard[r][c]=2;
    knightLocation[0]=r;
    knightLocation[1]=c;
    }
/*Provjera polja na koja moze stati skakac u 8 smjerova,
Na polja na koja moze stati adresa se upisuje u vektor freePosition,
a na polja na koja ne moze u vektor se upisuje broj sto
*/
void checkFreePosition(){
unsigned int k_row = knightLocation[0]; //skracivanje imena varijable
unsigned int k_col = knightLocation[1]; //skracivanje imena varijable
if((k_row-2)<8 && (k_col+1)<8 && chessBoard[k_row-2][k_col+1] == 1 ){
  freePosition[0]= (k_row-2)*10 + (k_col+1);
  } else {
        freePosition[0]= 100;
        }
```

```
if((k_row-1)<8 && (k_col+2)<8 && chessBoard[k_row-1][k_col+2] == 1 ){
  freePosition[1]= (k_row-1)*10 + (k_col+2);
  } else {
       freePosition[1]= 100;
       }
if((k_row+1)<8 && (k_col+2)<8 && chessBoard[k_row+1][k_col+2] == 1 ){
  freePosition[2]= (k_row+1)*10 + (k_col+2);
  } else {
       freePosition[2]= 100;
       }
if((k_row+2)<8 && (k_col+1)<8 && chessBoard[k_row+2][k_col+1] == 1 ){
  freePosition[3]= (k_row+2)*10 + (k_col+1);
  } else {
       freePosition[3]= 100;
       }
if((k_row+2)<8 && (k_col-1)<8 && chessBoard[k_row+2][k_col-1] == 1 ){
  freePosition[4]= (k_row+2)*10 + (k_col-1);
  } else {
       freePosition[4]= 100;
       }
if((k_row+1)<8 && (k_col-2)<8 && chessBoard[k_row+1][k_col-2] == 1 ){
  freePosition[5]= (k_row+1)*10 + (k_col-2);
  } else {
       freePosition[5]= 100;
       }
if((k_row-1)<8 && (k_col-2)<8 && chessBoard[k_row-1][k_col-2] == 1 ){
  freePosition[6]= (k_row-1)*10 + (k_col-2);
  } else {
       freePosition[6]= 100;
       }
if((k_row-2)<8 && (k_col-1)<8 && chessBoard[k_row-2][k_col-1] == 1 ){
  freePosition[7]= (k_row-2)*10 + (k_col-1);
  } else {
       freePosition[7]= 100;
       }
}
//Cisti polje na kojemu je skakac prethodno bio tj. upisuje 0 na tu adresu
void clean(){
    chessBoard[knightLocation[0]][knightLocation[1]] = 0;
}
/*Ispis adresa polja na koja skakac moze ici
te vraca informaciju o broju polja na koja se moze ici (0-8)
*/
unsigned int showFreePosition(){
cout<<endl<<"Mozete na polja: ";
unsigned int brojac = 0;
    for (int i=0; i<8; i++){
```

```
      if (freePosition[i] != 100){
       if (freePosition[i] < 10){
         cout<<"0"<<freePosition[i]<<" ";
       } else {
       cout<<freePosition[i]<<" ";
       }
       brojac++;
           }
   }
       cout<<endl<<endl;
       return brojac;
}
//Provjera dali je uneseni potez ispravan
bool checkMove(unsigned int cm){
    bool flag = 0;
    for (int i=0; i<8; i++){
       if (cm == freePosition[i] && cm != 100){  //ako je moguce napraviti potez vrati jedan
            flag = 1;
            }
       }
       return flag;
}
//Obrisi sve sa zaslona
void clearScreen(){
    system("CLS");
}
};
/*************************  START  SIMULACIJA  *************************/
void possible()
{
            int npos;
            for(int r=0;r<=7;r++)
            {
                        for(int c=0;c<=7;c++)
                        {
                                    npos = 0;
                                    for(int i=0;i<=7;i++)
                                    {
                                                if(((r+ver[i]      >=0)      &&      (r+ver[i]      <=7))&&((c+hor[i]      >=0)      &&      (c+hor[i]
<=7))&&(board[r+ver[i]][c+hor[i]] == 0))
                                                {
                                                            nextij[r][c][npos] = i;
                                                            npos++;
                                                }
                                    }
                                    accessible[r][c] = npos;
                        }
```

```
		}
}
void exits(int l)
{
		int min = accessible[row+ver[nextij[row][col][0]]][col+hor[nextij[row][col][0]]];
		int r = row+ver[nextij[row][col][0]],c=col+hor[nextij[row][col][0]];
		for(int i=1;i < accessible[row][col];i++)
			if(min >= accessible[row+ver[nextij[row][col][i]]][col+hor[nextij[row][col][i]]])
			{
				min =accessible[row+ver[nextij[row][col][i]]][col+hor[nextij[row][col][i]]];
				r = row + ver[nextij[row][col][i]];
				c = col + hor[nextij[row][col][i]];
			}
			board[r][c]=l;
			row = r;
			col = c;
}
/**************************   KRAJ SIMULACIJA   *************************/
//Glavni program
int main(int argc, char *argv[])
{
unsigned int x;     //pocetna pozicija skakaca
unsigned int next;  // sljedece polje
unsigned int y = 0; //brojac slobodnih polja
unsigned int z = 1; //brojac napravljenih poteza
unsigned int j = 0;
int choice = -1;
int count = 1;
int xx;
KnightTour kt;              //kreiraj objekt
kt.printInformation();      //ispisi podatke o autoru programa
//Ispisi izbornik na zaslon i trazi unos
do {
   cout<<endl<<" 1 - Igra skakacev obilazak"<<endl<<" 2 - Simulacija skakacevog obilaska"<<endl<<" 0 - Izlaz"<<endl<<endl;
   cout<<"Unesite vas odabir: ";
   cin>>choice;
   }while(choice < 0 || choice >2);
switch (choice){
     //Ako je unesena nula izadji iz programa
     case 0:
     system("PAUSE");
     return EXIT_SUCCESS;
     break;
     //Ako je unesen jedan obavi sve potrebne akcije kako bi se mogli igrati
     case 1:
     kt.clearScreen();       //ocisti zaslon
     kt.printInformation();  //ispisi podatke o autoru programa
```

```
    kt.newBoard();          //kreiraj novu praznu plocu
    kt.printBoard();        //ispisi praznu plocu na zaslon
    //Traziti od korisnika da unese pocetnu poziciju skakacaa sve dok ne unese broj od 00 do 77
            do{
              cout<<endl<<"Unesite pocetnu poziciju skakaca (npr. 52): ";
              cin>>x;
            } while (x < 0 || x > 77);
    kt.shiftKnight(x);        //staviti skakaca na pocetnu poziciju
    kt.checkFreePosition();  //provjeri polja na koja skakac moze ici
    kt.clearScreen();       //ocisti zaslon
    kt.printInformation();   //ispisi podatke o autoru programa
    kt.printBoard();        //prikazi plocu sa ucrtanim skakacem na zaslon
                    do{
                      y = kt.showFreePosition();  //preuzmi broj polja na koja skakac moze stati i pohrani u varijablu y
                      if (y == 0){              //ako nema vise slobodnih mjesta, prekinu do-while petlju
                      break;
                      }
                     z++;     //povecaj broj poteza za jedan
                           //Traziti od korisnika unos sljedece pozicije skakaca sve dok ne unese polje na koje kralj moze stati
                           do{
                             cout<<"Unesite #"<<z<<" poziciju skakaca: ";
                             cin>>next;
                             j = kt.checkMove(next); //provjeri dali se na uneseno polje moze stati
                           } while (j == 0);
                     kt.clean();              //na polje na kojemu je skakac bio upisi nulu
                     kt.shiftKnight(next);       //pomaci skakaca na uneseno polje
                     kt.checkFreePosition();    //provjeri sljedece slobodne pozicije na koje moze skakac stati
                     kt.clearScreen();         //ocisti zaslon
                     kt.printInformation();     //ispisi podatke o autoru programa
                     kt.printBoard();          //ispisi plocu na zaslon
                    } while (y > 0);
    //ako se ploca ne obidje u 64 poteza (na svako polje stanemo jednom) dobijemo informaciju o kraju igre
        if(z != 64){
            cout<<"        KRAJ IGRE!!!!            "<<endl;
            cout<<"Nazalost niste uspjeli doci do kraja."<<endl;
            cout<<"Odmorite skakaca pa pokusajte ponovno!"<<endl<<endl;
        } else {
              cout<<"BRAVO!!! USPJESNO STE OBISLI PLOCU!!!"<<endl<<"CESTITAMO!!!"<<endl<<endl;
             }
    system("PAUSE");
    return EXIT_SUCCESS;
    break;
    //Ako je unesen broj dva traziti unos pocetne pozicije skakaca i dalje prikazati simulaciju obilaska
    case 2:
            int count = 1;
//Traziti od korisnika da unese pocetnu poziciju skakaca sve dok ne unese broj od 00 do 77
        do{
```

```
          cout<<endl<<"Unesite pocetnu poziciju skakaca (npr. 52): ";
          cin>>xx;
         } while (xx < 0 || xx > 77);
          row = xx / 10;
          col = xx % 10;

             board[row][col]=count;   //postavi skakaca na pocetnu poziciju, koristi pomocnu plocu - board
             while(count!=64)
             {
                        count++;
                        possible();
                        exits(count);
             }
KnightTour kt2;       //kreiraj objekt
kt2.newBoard();     //kreiraj novu praznu plocu
int position[64];       //pozicije skakaca
//Obilazak ploce
for(int i = 0; i < 64; i++){
  for(int j = 0; j<8; j++){
     for (int k = 0; k < 8; k++){
        if(board[j][k] == i+1){
          position[i] = (j*10) + k;     //pretvaranje dvodimenzionalne kordinate u jednodimenzionalnu
          kt2.clearScreen();        //ocisti zaslon
          kt2.shiftKnight(position[i]);   //pomici skakaca prema redosljedu (1-64)
          kt2.printInformation();    //ispisi podatke o autoru programa
          kt2.printBoard();         //ispisi plocu sa ucrtanim skakacem
          kt2.clean();              //ocisti polje na kojemu se nalazio skakac
          cout<<endl<<"Trenutna pozicija skakaca: ";
          if (position[i] < 10){
            cout<<"0"<<position[i]<<endl;         //ako je adresa polja manja od 10 dodaj ispred nulu
          } else {
            cout<<position[i]<<endl;              //inace ispisi adresu
          }
          Sleep(1000);                        //cekaj 1000 milisekundi
        }
     }
  }
}
cout<<endl<<"Ploca je obidjena sljedecim redosljedom:"<<endl<<endl;
for (int i = 0; i < 64; i++){
//    cout<<"#"<<i+1<<": ";
   if (position[i] < 10){
      cout<<"0"<<position[i]<<endl;
   } else {
        cout<<position[i]<<endl;
        }
}
```

```
//cout<<positions[0]<<endl<<positions[1]<<endl;;
/* Ispisi rjesenje obilaska
for(int j=0;j<=7;j++)
        {
                for(int k=0;k<=7;k++)
                        cout <<  setw(3) << board[j][k];
                cout <<"\n\n";
        } */
    system("PAUSE");
    return EXIT_SUCCESS;
    break;
    }
}
```

LITERATURA

[1.]http://en.wikipedia.org/wiki/Knight%27s_tour

[2.]http://blog.tiaan.com/link/2010/06/01/knights-tour-warnsdorff-csharp-cplusplus

[3.]http://www.cprogramming.com/compiler.html