Mateusz Zacharecki

Opis programu “Konik szachowy”

Dane: liczba naturalna N
Wynik: Droga konika szachowego na szachownicy N x N, która odwiedza każde pole dokładnie jeden raz.

Deklaruję istotne zmienne, gdzie N to wymiar szachownicy N x N, tablica h będzie naszą szachownicą, a w tablicach a,b zapamiętamy różnice współrzędnych:

```
#define N 5
int h[N][N], a[8], b[8];
```

Funkcja wypisz typu void ma za zadanie wypisać naszą szachownicę w sposób graficzny. U nas wygląda ona tak:

```
void wypisz(int g[][N])
{
   int i,j;
   for (i=0;i<N;i++){
      for (j=0;j<N;j++){
      printf("%d ",g[i][j]);
      }
      printf("\n");
   }
}
```

Kluczowa dla działania naszego programu jest funkcja skok typu integer. Deklarujemy zmienne lokalne pozycjax, pozycjay typu integer, za pomocą których będziemy oznaczać współrzędne następnego pola w szachownicy. Przy każdej próbie wykonania następnego ruchu będziemy zwiększać zmienną licznik typu integer o 1. Zmienna ta posłuży nam, żeby przy udanej próbie zapamiętać dla bieżącej pozycji skoczka, na którym z kolei polu znajduję się w danym momencie skoczek. Numer ten zapisany zostanie w tablicy h. Do określenia bieżącej pozycji skoczka posłużą nam zmienne x,y. Jeżeli już przeszliśmy skoczkiem przez wszystkie pola szachownicy zwracamy prawdę. W przeciwnym przypadku zapoczątkowujemy wybór nowych kandydatów. Do obliczenia współrzędnych następnego pola w szachownicy w zależności od x i y będziemy dodawać różnice współrzędnych zapamiętane uprzednio w tablicach a i b. Jeżeli współrzędne następnego pola w szachownicy nie będą wykraczać poza zakres szachownicy oraz jeżeli współrzędne następnego pola w szachownicy nie były już wcześniej zajmowane przez skoczka to wykonujemy skok. Jeżeli ruch ten jest możliwy do wykonania to funkcja zwraca prawdę. W przeciwnym wypadku usuwamy kandydata z częściowego rozwiązania i zwracamy fałsz.

```
int próbuj wykonać następny ruch
{
   zwiększ o 1 numer kolejnego potencjalnego ruchu;
   zapamiętaj mój potencjalny ruch w tablicy;
   if (konik przeszedł przez wszystkie pola szachownicy, tzn. Jeżeli został wykonany N*N-ty ruch)
      zwróć prawdę;
   zapoczątkuj wybieranie kandydatów
```

```
   {
      obierz potencjalne miejsce skoku;
      if (mogę wykonać skok na nowe potencjalne pole)
      {
         if (ruch jest możliwy do wykonania)
         {
            zwróć prawdę;
         }
      }
   }
   usuń kandydata z częściowego rozwiązania;
   zwróć fałsz;
}
```

Implementacja pomysłu w C:

```
int skok(int x, int y, int licznik){
   int pozycjax,pozycjay;
   licznik +=1;
   h[x][y] = licznik;
   if (licznik == N*N)
      return 1;
   for(int i=0;i<8;i++){
         pozycjax= x + a[i];
         pozycjay= y + b[i];
      if(pozycjax >= 0 && pozycjax < N && pozycjay >= 0 && pozycjay < N &&
h[pozycjax][pozycjay]==0){
         if(skok(pozycjax,pozycjay,licznik))
         {
            return 1;
         }
      }
   }
   h[x][y] = 0;
   return 0;
}
```

W funkcji main typu integer wprowadzam możliwe ruchy skoczka po szachownicy. Następnie w każdym miejscu tablicy h wpisuję 0, jedynie na pozycji początkowej skoczka wpisuję 1. Jeżeli procedura skok zwróci prawdę, program wykonuje procedurę wypisz(h), w przeciwnym wypadku program wyświetla komunikat “Nie ma rozwiązań”. Funkcja main będzie wyglądać tak:

```
int main()
{
   a[0]=2; b[0]=1;
   a[1]=1; b[1]=2;
   a[2]=-1; b[2]=2;
   a[3]=-2; b[3]=1;
```

```
    a[4]=-2; b[4]=-1;
    a[5]=-1; b[5]=-2;
    a[6]=1; b[6]=-2;
    a[7]=2; b[7]=-1;
    int i,j;
    for (i=0;i<N;i++){
        for (j=0;j<N;j++){
        h[i][j]=0;
        }
    }
    h[0][0]=1;
    if (skok(0,0,0)) wypisz(h);
    else printf("Nie ma rozwiazan!");
    return 0;
}
```