DO NOWEJ PODSTAWY PROGRAMOWEJ

Koło informatyczne dla szkół ponadgimnazjalnych

Informatyka Europejczyka

Wiesława Amietszajewa

Helion EDUKACJA

Redaktor prowadzący: Tomasz Waryszak
Projekt okładki: ULABUKA

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/iekolp
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Dodatkowe materiały do książki można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/iekolp.zip

ISBN: 978-83-246-6615-7

Printed in Poland.

Spis treści

1.11. Elementy teorii gier

Definicja

Teoria gier to dział matematyki zajmujący się badaniem optymalnych zachowań w przypadku sytuacji konfliktowych. Celem gracza w każdej grze jest oczywiście wygrana, która łączy się z osiągnięciem korzyści (zebranie skarbów, zdobycie władzy czy po prostu przetrwanie, przeżycie). Teorie gier mają zastosowanie w wielu dziedzinach. Ich zadaniem jest badanie strategii, jakie powinien przyjąć gracz, żeby osiągnąć jak najlepszy wynik.

Przy opracowywaniu strategii zawsze należy brać pod uwagę:
- zbiór graczy,
- reguły,
- możliwe straty,
- możliwe wyniki,
- wypłaty (korzyści).

Zajmiemy się analizą popularnych problemów w teorii gier — grą w życie i grą logiczną NIM.

1.11.1. Gra w życie

W drugiej połowie XX wieku John Conway wymyślił grę, która zdobyła bardzo dużą popularność. Gra jest modelem rodzenia się, ewolucji, śmierci kolonii organizmów. Rozgrywa się w przestrzeni (dwuwymiarowa plansza podzielona na komórki) i czasie (kolejne pokolenia).

Grę rozpoczynamy od „zasiedlenia" komórek.

Zasady gry według Conwaya:
- Martwa komórka, która ma trzech sąsiadów, rodzi się.
- Żywa komórka pozostaje żywa, jeżeli ma dwóch lub trzech sąsiadów.

- Jeżeli żywa komórka ma mniej niż dwóch sąsiadów, umiera z samotności.
- Jeżeli żywa komórka ma więcej niż trzech sąsiadów, umiera z zatłoczenia.

Do komputerowej realizacji tego algorytmu wykorzystamy tablicę, która będzie symulować kolonię żywych organizmów. Komórki niezamieszkałe zapełniamy literą „o", pojawienie się organizmu zaznaczamy literą „x". Potrzebna nam jest druga plansza, która przechowuje stan kolejnego pokolenia. Zawartości kolonii nie możemy zmieniać dynamicznie, bo przecież procesy „rodzenia się" i „umierania" zachodzą w tym samym czasie.

Wskazówka

Tworzymy dwie plansze o rozmiarze stosownym do symulacji, którą chcemy przeprowadzić. Wypełniamy je na początku „o" (oczyszczamy z intruzów ☺). Kontrolnie możemy wyświetlić, że wszystko jest gotowe.

```
char plansza[20][20];
char nowa_plansza [20][20];
int i,j,w,k,ilosc=0,pokolenie,p=0;
for( i=0;i<20;i++)
 { for ( j=0;j<20;j++)
  { plansza[i][j]='o';
    nowa_plansza [i][j]='o';
    cout<<plansza[i][j]<<" ";
  }
```

Następnie zasiedlamy kolonię, podając współrzędne wiersza (`w`) i kolumny (`k`), gdzie ma „pojawić się życie". Przy wyborze miejsc do zasiedlenia pamiętajmy, żeby w żadnym pokoleniu nie przekroczyć zakresu tablicy. Zmienna ilość określa, ile mamy żywych organizmów do zasiedlenia. Wyświetlimy sobie od razu schemat naszej kolonii.

```
cin>>ilosc;
for (int l=0; l<ilosc; l++)
{cin>>w;
 cin>>k;
 plansza[w][k]='x';
}
cout<<"Kolonia: "<<endl;
for( i=0;i<20;i++)
 { for (int j=0;j<20;j++)
   cout<<plansza[i][j]<<" ";
 cout<<endl;
}
```

Wskazówka

Pozostało nam napisanie fragmentu kodu, który zawiera zasady gry. Żeby nie „wyskoczyć" poza planszę, proponuję zacząć od indeksów równych 1 i skończyć na 15. Oczywiście można uzupełnić program o funkcję informującą, czy w aktualnym rozmiarze tablicy możliwy jest dalszy rozwój, do czego zachęcam.

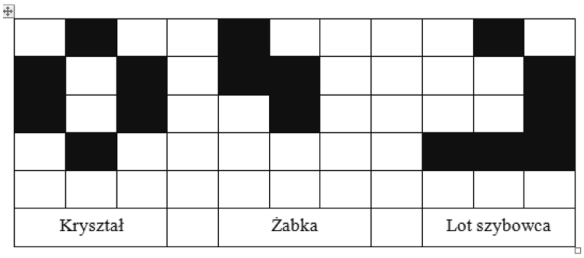
Poniższy fragment kodu wykona symulację dla jednego pokolenia. Jeżeli chcemy symulacji dla *n*-tego, należy wywołać go *n*-krotnie.

```
for (i=1; i<15;i++)
 for (j=1; j<15; j++)

//Zmienna zliczająca liczbę sąsiadów
  {ilosc=0;
  //Pojawia się nowe życie
  if (plansza[i-1][j-1]=='x') ilosc++;
  if (plansza[i-1][j]=='x') ilosc++;
  if (plansza[i-1][j+1]=='x') ilosc++;
  if (plansza[i][j-1]=='x') ilosc++;
  if (plansza[i][j+1]=='x') ilosc++;
  if (plansza[i+1][j-1]=='x') ilosc++;
  if (plansza[i+1][j]=='x') ilosc++;
  if (plansza[i+1][j+1]=='x') ilosc++;
  if (ilosc==3) nowa_plansza[i][j]='x';
  //Nic się nie zmienia
if (ilosc==2) nowa_plansza[i][j]=plansza[i][j];
//Organizm umiera z zatłoczenia
  if (ilosc>3) nowa_plansza[i][j]='o';
//Organizm umiera z samotności
 if (ilosc<2) nowa_plansza[i][j]='o';
 }
//Funkcja kopiująca struktury, w konkretnym przypadku tablicę
//„nowa_plansza" do „plansza" (kolejne pokolenie jest bazowym
//dla następnego). Trzecim argumentem jest funkcja określająca
//liczbę kopiowanych elementów
memcpy(plansza, nowa_plansza, sizeof(plansza));
```

Mamy już wszystkie fragmenty gry. Pozostaje połączyć je w całość. Powodzenia!

Zadanie 1.28 Wykorzystując napisany program, sprawdź, jak będzie wyglądać kolonia bakterii w piątym pokoleniu dla struktur przedstawionych na rysunku 1.17.



Rysunek 1.17. Kolonie bakterii

Gra w życie jest przykładem automatu komórkowego. Automaty komórkowe stanowią już właściwie odrębny dział nauki i znajdują wiele zastosowań. Czy potrafisz wymienić przykłady?

1.11.2. Gra logiczna NIM

Czy znasz zabawę polegającą na zabieraniu przedmiotów (np. zapałek, kamieni) z kilku stosów (rzędów), taką że na starcie każdy miał różną liczbę elementów? W myśl zasad gry, wykonując ruch, gracz może zabrać dowolną liczbę elementów, ale tylko z jednej sterty. Ten, kto zabiera ostatni element, wygrywa lub przegrywa, w zależności od przyjętej koncepcji.

Zaimplementowanie takiej gry do komputera wydaje się banalne. Przeanalizuj umieszczony niżej przykład dla trzech rzędów po maksymalnie 10 elementów. Liczbę elementów do każdego rzędu losuje komputer. Podczas wyświetlania elementy są symbolizowane znakiem @. W implementacji zakładamy, że gracz potrafi liczyć i nie zabiera z rzędu więcej elementów, niż się w nim znajduje ☺

Gra NIM dla dwóch graczy; wygrywa osoba, która weźmie ostatni kamyczek

```
#include <iostream>
#include <cstdlib>
using namespace std;
void wyswietl(int i1,int i2,int i3)
{int i;
 for ( i=1; i<=i1; i++)
    cout<<"@"<<" ";
    cout<<endl;
 for (i=1; i<=i2; i++)
     cout<<"@"<<" ";
    cout<<endl;
   for (i=1; i<=i3; i++)
    cout<<"@"<<" ";
    cout<<endl;
 }
```

```
int main ()
{int rzad1[10],rzad2[10],rzad3[10];;
 int rzad, ilosc;
 int i,i1=0,i2=0,i3=0,gracz1,gracz2,numer=0;
 srand(time(NULL));
 for (i=0; i<10; i++)
  {rzad1[i]=rand()%2;
   if (rzad1[i]==1) i1++;
   }
 for (i=0; i<10; i++)
  {rzad2[i]=rand()%2;
   if (rzad2[i]==1) i2++;
   }
 for (i=0; i<10; i++)
   {rzad3[i]=rand()%2;
    if (rzad3[i]==1) i3++;
   }
   wyswietl(i1,i2,i3);

 while ((i1+i2+i3)>0)
  {numer++;
 if (numer%2==0) cout<<"Gracz 2"<<endl;
 else cout<<"Gracz 1"<<endl;
 cout<<"podaj numer wiersza i ile zabierasz"<<endl;
 cin>>rzad;
 cin>>ilosc;
 if (rzad==1) i1=i1-ilosc;
 if (rzad==2) i2=i2-ilosc;
 if (rzad==3) i3=i3-ilosc;
 wyswietl(i1,i2,i3);
 if (i1+i2+i3==0) cout<<"Wygrana!!"<<endl;
}
 system("pause");
 return 0;
}
```

Przeprowadź symulację (możesz oczywiście zmienić liczbę wierszy i kamyków) i zastanów się nad zwycięską strategią. Czy potrafisz napisać program, w którym jednym z grających jest komputer?

W klasycznej grze wygrywa ten, kto zabierze ostatni element. Strategią wygrywającą jest ta, w której suma wartości elementów (dodawanie nimliczb) jest równa 0. Tylko... co to są te nimliczby?

Wskazówka

Nimliczby — liczby, które różnią się od liczb naturalnych sposobem wykonywania działań.

1.11.3. Dodawanie nimliczb

- Suma dwóch równych nimliczb wynosi 0: 7⊕7 = 0.
- Jeżeli większa z nimliczb jest potęgą dwójki, dodaje się je jak zwykłe liczby: 8⊕7 = 15.

Dodawanie nimliczb odpowiada operacji XOR (w C++ operator ^) na cyfrach rozwinięcia dwójkowego danej liczby.

Dodajmy dwie liczby:

$37_D = 100101_B$

$51_D = 110011_B$

Wskazówka

Wynikiem operacji XOR jest prawda wtedy, gdy oba elementy są różne
(1 XOR 1 = FAŁSZ, 0 XOR 0 = FAŁSZ, 0 XOR 1 = PRAWDA, 1 XOR 2 = PRAWDA).

XOR	100101_B	37_D
	110011_B	51_D
NIM — suma	010110_B	22_D

37⊕51 = 22

Powyższe działanie w C++ zapisujemy tak:

```
wynik = 37 ^ 51;
```

1.11.4. Mnożenie nimliczb

- Jeżeli większa z nimliczb jest równa elementowi ciągu 1, 2, 4, 16, 256, …, to mnożenie odbywa się na zwykłych zasadach.
- Jeżeli nimliczbę mnoży się przez siebie, to wynik jest równy sumie dwóch nimliczb — jej samej i części całkowitej jej połowy: 9⊙9 = 9⊕9/2 = 9⊕4 = 13.

1.11.5. Potęgowanie nimliczb

- Potęgowanie odbywa się na standardowych zasadach — poprzez wielokrotne mnożenie nimliczb, czyli:

$$x^3 = x^*\xi^*x.$$

Zadanie 1.29 Napisz program — kalkulator wykonujący operacje dodawania, mnożenia i potęgowania nimliczb. Korzystanie z niego podczas gry w NIM ułatwi wygrywanie ☺

Powróćmy do strategii wygrywającej. Mamy trzy rzędy elementów.

Numer rzędu	Elementy	Ilość w systemie dziesiętnym	Ilość w systemie binarnym
1	I I I I I	5	101
2	I I I I I I I	7	111
3	I I I	3	011
NIM suma (operacja XOR)		5^7^3 = 1	001

Żeby NIM suma była równa 0, wystarczy z dowolnego rzędu zabrać jeden element. Znajdziemy się wtedy na pozycji wygrywającej.

Istnieje ryzyko, że liczba elementów do zabrania przekracza liczbę elementów w największym stosie. Jak w tym przykładzie.

Numer rzędu	Elementy	Ilość w systemie dziesiętnym	Ilość w systemie binarnym
1	I I I I I I I I I	9	1001
2	I I I I I I I	7	0111
3	I I I	3	0011
NIM suma (operacja XOR)		9^7^3 = 13	1101

W takiej sytuacji należy od otrzymanej liczby odjąć liczbę elementów w największym stosie i zostawić na nim liczbę elementów równą otrzymanej różnicy:

13–9 = 4

Zabieramy 9–4 = 5

Numer rzędu	Elementy	Ilość w systemie dziesiętnym	Ilość w systemie binarnym
1	I I I I	4	100
2	I I I I I I I	7	111
3	I I I	3	011
NIM suma (operacja XOR)		4^7^3 = 0	000

FTP **Zadanie 1.30** Napisz program pt. „Gra NIM” dla trzech stosów po 10 elementów. Jednym z grających jest komputer.

Zadanie 1.31 Zmień w swoim programie warunek wygranej — ten, kto zabiera ostatni element, przegrywa.

Informatyka bez tajemnic, czyli jak przerosnąć mistrza

Nie można oderwać Cię od monitora? Inni pytają Cię, co w sieci piszczy? Zastanawiasz się, jak używać komputera przy wypełnianiu Twoich codziennych obowiązków? Jeśli tak, lekturę czas zacząć — i ruszyć na podbój świata rządzącego się zero-jedynkowymi prawami.

- **Dowiedz się, jak przekazywać informacje komputerom za pomocą algorytmów.**
- **Zostań mistrzem arkuszy kalkulacyjnych, a może nawet — w przyszłości — wielkim finansistą.**
- **Rozgryź tajniki darmowego systemu operacyjnego Linux.**
- **Stań się prawdziwym sieciowym guru!**

Zmuś komputer do pracy według Twoich wytycznych dzięki opanowaniu jego sposobu myślenia! Poszerz swoją wiedzę o kluczowe zagadnienia z informatyki i wykorzystaj ją w praktyce.

Z książką do dodatkowych zajęć komputerowych z serii ***Informatyka Europejczyka*** uczniowie szkół ponadgimnazjalnych będą systematycznie poszerzać swoją wiedzę informatyczną. Pomoże im to w przygotowaniach do matury z informatyki oraz zachęci do udziału w konkursach. Książka zawiera wiele zestawów ćwiczeń i krótkich zadań z rozwiązaniami, które nie powielają materiału przerabianego na lekcjach.

Spraw, aby to komputery służyły Tobie i spełniały Twoje zachcianki!

Wciśnij Enter i do dzieła!

http://edukacja.helion.pl

helion.pl księgarnia internetowa

Nr katalogowy: 12416

Księgarnia internetowa: http://helion.pl

Zamówienia telefoniczne: 0 801 339900 0 601 339900

Helion EDUKACJA

Sprawdź najnowsze promocje: http://helion.pl/promocje
Książki najchętniej czytane: http://helion.pl/bestsellery
Zamów informacje o nowościach: http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

ISBN 978-83-246-6615-7

9 788324 666157

Informatyka w najlepszym wydaniu