Міністерствоосвіти і науки України

Національний університет „Львівська політехніка”

Кафедра ЕОМ



**Лабораторна робота №3**

З дисципліни:”Паралельні та розподіленні обчислення ”

На тему:”Можливості використання паралельних алгоритмів.”

Виконав: ст.гр. КІ-33

Мельник А.О.

Прийняв: Козак Н.Б.

Львів 2020

Мета роботи: Дослідити можливості розв’язання різноманітних задач за допомогою паралельних алгоритмів. Навчитися виділяти незалежні гілки обчислень та виконувати їх паралельно.

Варіант №11

*Завдання:*

|  |  |
| --- | --- |
| 5,  11 | Запропонувати та відобразити алгоритм реалізації гри “LIFE”. Гра моделює життя деякої колонії живих клітин, які виживають, розмножуються або гинуть згідно наступних умов. Клітина виживає, якщо вона має двох або трьох сусідів з восьми можливих. Якщо у клітини лише один сусід, або немає жодного, то вона гине (від ізоляції). Якщо клітина має більше трьох сусідів, то вона гине (від перенаселення). В будь-якій порожній позиції, яка має рівно трьох сусідів у наступному поколінні з’являється нова клітина. Гра повинна починатися з довільної кількості клітин, що розташовані в ігровому полі випадковим чином (інтерактивний режим задавання вхідних даних) – так звана початкова популяція. Програма повинна коректно завершувати роботу у таких випадках: а).загинула вся популяція, б) на вимогу користувача. |

*Аналіз задачі та опис незалежних подій.*

Після аналізу завдання в ньому можна виділити гілки: задання початкової колонії, обчислення поточного елементу, вивід колонії. Виділивши ці гілки, оформляємо їх як окремі функції і використовуємо в програмі.

**Лістинг програми:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include "stdio.h"

#include "stdlib.h"

#include "conio.h"

int\*\* inp\_mtx(int);

void rand\_mtx(int\*\*, int, int);

void out\_mtx(int, int\*\*);

void f\_out\_mtx(FILE\*, int, int\*\*);

void main()

{

setlocale(LC\_ALL, "Ukrainian");

int n, ch, ch1, a, b, i, j, x, it = 0;

int alive = 0, neighbors = 0;

int\*\* coloniya;

FILE\* fptr;

if ((fptr = fopen("D://rezult.txt", "w")) == NULL)

{

printf("\nНе можу відкрити файл");

exit(1);

}

printf("Введення розмiру колонii\n");

printf("розмiр = ");

scanf("%d", &n);

coloniya = inp\_mtx(n);

printf("--------------------------------------\n");

printf("\nВведiть спосiб задання колонii:\n");

printf(" Вручну- натиснiть 1\n");

printf(" Автоматично- натиснiть 0\n");

printf("Зробiть вибiр [1/0]... ");

scanf("%d", &ch);

printf("--------------------------------------\n");

if (!ch)

{

printf("\nВведiть будь-яке число вiд %d до %d: ", n + 10, n \* n);

scanf("%d", &x);

rand\_mtx(coloniya, n, x);

}

else

{

printf("\nВведiть позицii, де будуть знаходитись клiтини\n");

printf("(наприклад: 3 3, 7 1, 2 5...)\n");

while (ch != 'n')

{

scanf("%d%d", &a, &b);

coloniya[a][b] = 1;

alive++;

printf("продовжити? [y/n] ");

ch = \_getche();

printf("\n");

}

}

printf("--------------------------------------\n");

printf("\nПочаткова колонiя\n");

out\_mtx(n, coloniya);

printf("--------------------------------------\n\n");

\_getch();

alive = 0;

do

{

alive = 0;

for (i = 1; i <= n; i++)

{

for (j = 1; j <= n; j++)

{

if (coloniya[i - 1][j] == 1)

neighbors++;

if (coloniya[i - 1][j + 1] == 1)

neighbors++;

if (coloniya[i][j + 1])

neighbors++;

if (coloniya[i + 1][j + 1] == 1)

neighbors++;

if (coloniya[i + 1][j])

neighbors++;

if (coloniya[i + 1][j - 1] == 1)

neighbors++;

if (coloniya[i][j - 1])

neighbors++;

if (coloniya[i - 1][j - 1] == 1)

neighbors++;

if (coloniya[i][j])

{

if (neighbors <= 1)

coloniya[i][j] = 0;

if (neighbors > 3)

coloniya[i][j] = 0;

if ((neighbors == 2) || (neighbors == 3))

coloniya[i][j] = 1;

}

else

{

if (neighbors == 3)

coloniya[i][j] = 1;

else coloniya[i][j] = 0;

}

if (coloniya[i][j])

alive++;

fprintf(fptr, "i=%d j=%d сусiди=%d живi=%d\n", i, j,

neighbors, alive);

f\_out\_mtx(fptr, n, coloniya);

neighbors = 0;

}

}

it++;

out\_mtx(n, coloniya);

printf("Iтерацiя %d, кiлькiсть живих клiтин %d\n", it, alive);

printf("--------------------------------------\n\n");

fprintf(fptr, "Iтерацiя %d, кiлькiсть живих клiтин %d\n", it, alive);

fprintf(fptr, "--------------------------------------\n\n");

ch1 = \_getch();

if (ch1 == 'e')

break;

} while (alive != 0);

printf("Вихiд з програми\n");

fclose(fptr);

}

int\*\* inp\_mtx(int rozm)

{

int i, \*\* m;

m = (int\*\*)calloc((rozm + 2), sizeof(int\*));

for (i = 0; i < (rozm + 2); i++)

m[i] = (int\*)calloc((rozm + 2), sizeof(int));

return m;

}

void out\_mtx(int rozm, int\*\* m)

{

int i, j;

for (i = 1; i <= rozm; i++)

{

for (j = 1; j <= rozm; j++)

printf("%3d ", m[i][j]);

printf("\n\n");

}

printf("\n");

}

void f\_out\_mtx(FILE\* fp, int rozm, int\*\* m)

{

int i, j;

for (i = 1; i <= rozm; i++)

{

for (j = 1; j <= rozm; j++)

fprintf(fp, "%3d ", m[i][j]);

fprintf(fp, "\n\n");

}

fprintf(fp, "\n");

}

void rand\_mtx(int\*\* mtx, int rozm, int max)

{

int i;

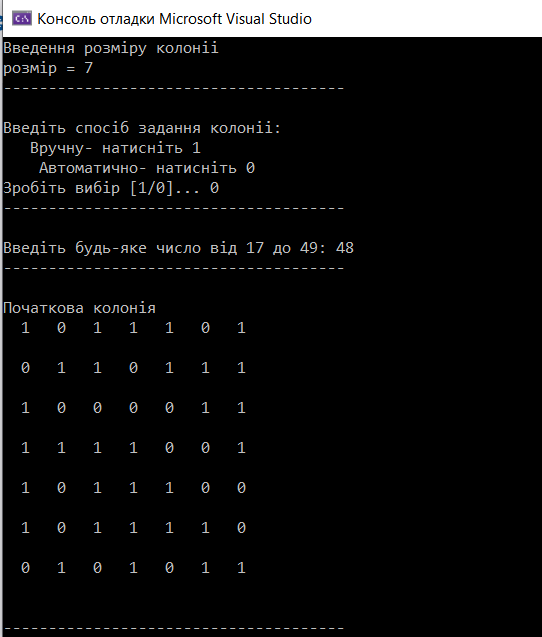
for (i = 0; i < max; i++)

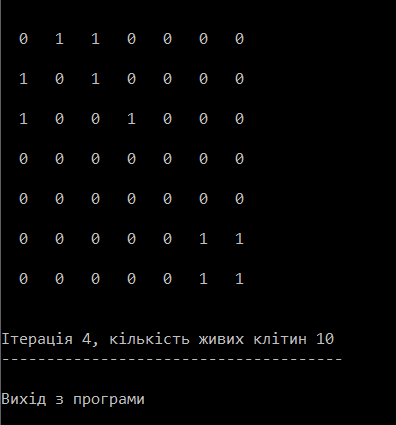
mtx[1 + rand() / (RAND\_MAX / rozm)][1 + rand() / (RAND\_MAX / rozm)] = 1;

}

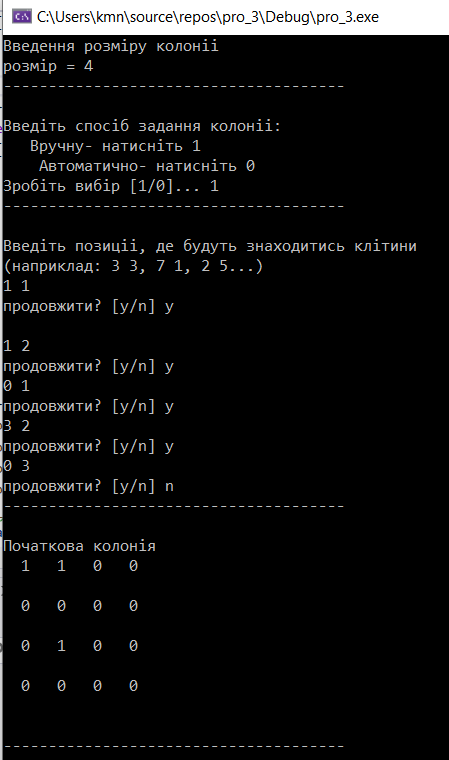
**Результати виконання роботи:**

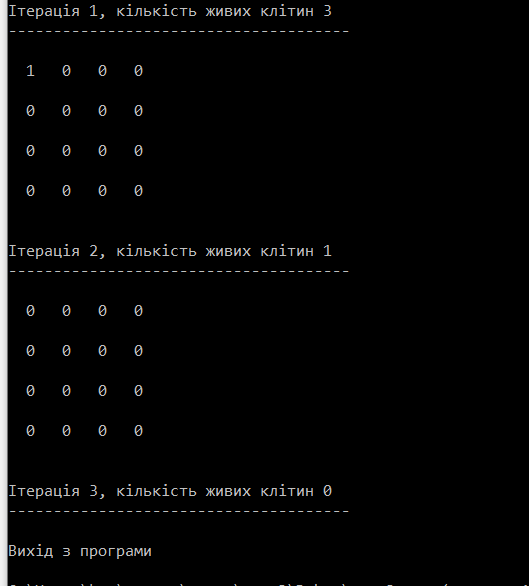
1. Автоматичне створення початкової колонії, вихід з програми на вимогу користувача





2) Задання початкової колонії вручну, вихід з програми- вся популяція загинула





**Висновки:** виконуючи дану лабораторну роботу я дослідив можливості розв’язання різноманітних задач за допомогою паралельних алгоритмів, а також застосував паралельне представлення алгоритмів на практиці. Створена мною програма використовує такий спосіб паралелізму, як розбиття задачі на підзадачі, а також демонструє можливість використання паралельних обчислень.