**Міністерство освіти і науки України**

**Національний університет «Запорізька політехніка»**

Кафедра програмних засобів

**Програмна документація**

до програми «Гра життя»

на мові програмування С++

розробленої студентами КНТ-114

Шкуріним Ігорем та Андрієм Жариком

для самостійної роботи №2

з дисципліни «Soft skills, групова динаміка та комунікації»

**Вміст:**

1. [Технічне завдання](#techtask)
2. [Специфікація](#Specification)
3. [Опис програми](#Programopys)
4. [Текст програми](#Programmtext)
5. [Керівництво програміста](#Programkerivnyctvo)
6. [Керівництво системного програміста](#sysprogkerivnyctvo)
7. [Керівництво оператора](#operkerivnyctvo)

**2025**

**ТЕХНІЧНЕ ЗАВДАННЯ**

1. Вступ:

Програмний засіб «Гра життя» реалізує клітинний автомат, в якому моделюється життя клітин на прямокутному полі за заданими правилами. Область застосування – навчальна та демонстраційна: програмний продукт використовується у рамках вивчення алгоритмів, логіки програмування та імітаційного моделювання. Об’єкт використання – комп’ютерна система студента/користувача.

1. Підстави для розробки:

* Самостійна робота з курсу “Soft skills, групова динаміка та комунікації”
* Освітній заклад: Національний університет «Запорізька політехніка»
* Затверджено на засіданні кафедри програмних засобів, дата: 30 березня 2023 р.

1. Призначення розробки

Програмний засіб призначений для моделювання клітинного автомата за правилами «Гри життя» Джона Конвея, дозволяє візуалізувати етапи еволюції клітинного поля залежно від початкової конфігурації, що задається користувачем.

1. Вимоги до програми чи програмного виробу

* Введення розмірів поля користувачем;
* Ініціалізація живих клітин вручну;
* Виведення стану поля на кожному кроці;
* Застосування правил гри для формування нового покоління;
* Автоматичне завершення гри, якщо на полі не залишилось живих клітин.

1. Вимоги до програмної документації

* вступ;
* підстави для розробки;
* призначення розробки;
* вимоги до програми чи програмному виробу;
* вимоги до програмної документації;
* техніко-економічні показники;
* стадії та етапи розробки;
* порядок контролю та приймання [15].

1. Техніко-економічні показники

* Очікувана ефективність: підвищення якості засвоєння знань студентами
* Річна потреба: використання у межах навчального курсу
* Переваги: безкоштовне рішення, яке не вимагає додаткових витрат на ліцензії.

1. Стадії та етапи розробки

* Постановка задачі.
* Розробка алгоритму.
* Програмування функціоналу.
* Тестування.
* Підготовка документації.

1. Порядок контролю та приймання

Програмний продукт вважається прийнятим у разі: правильного виконання всіх функцій згідно з описом, відсутності помилок при тестуванні типових сценаріїв, наявності повного комплекту документації;

**СПЕЦИФІКАЦІЯ**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № з/п | Позначення | Найменування   |  | | --- | |  | | Примітка | |
| Документація | | | | |
| 1 | ГЖ.00.00.00 ТЗ | Технічне завдання | | - |
| 2 | ГЖ.00.00.00 КО | Керівництво оператора | | - |
| 3 | ГЖ.00.00.00 КП | Керівництво програміста | | - |
| Комплекси | | | | |
| 4 | ГЖ.01.00.00 СП | Програмний комплекс «Гра життя». | | - |
| Компоненти | | | | |
| 5 | ГЖ.01.01.00 ІД | Ініціалізація даних. | | - |
| 6 | ГЖ.01.02.00 ВП | Виведення поля. | | - |
| 7 | ГЖ.01.03.00 КС | Крок симуляції. | | - |
| 8 | ГЖ.01.04.00 КП | Контроль перезапуску гри | |  |
| 9 | ГЖ.01.05.00 КК | Контроль завершення гри. | | - |

**ОПИС ПРОГРАМИ**

1. Загальні відомості

* Позначення програми: ГЖ.00.01 ПП
* Найменування: «Гра життя»
* Програмне забезпечення: компілятор C++ (MinGW), середовище виконання Windows.
* Мова програмування: C++

1. Функціональне призначення

Програма призначена для симуляції клітинного автомата «Гра життя», запропонованого Джоном Конвеєм.

1. Опис логічної структури

Алгоритм:

* Користувач задає розміри поля.
* Вводиться початкова конфігурація (живі клітини).
* Підраховується кількість живих сусідів для кожної клітини.
* Оновлюється стан поля згідно з правилами гри:
* Повторюється, поки поле не стане повністю мертвим.
* Запитує, чи не бажає користувач перезапустити гру.

Структура:

* fieldcreate(); - створення початкового поля
* initConfig(); - ініціалізація початкової конфігурації живих клітин користувачем
* gameLoop(); - основний цикл гри
* fielddelete(); - видалення поля
* askRestart(); - запит перезапуску гри

1. Використані технічні засоби

* Комп'ютери: IBM PC-сумісні, архітектура x86/x64.
* Операційні системи: Windows 11

1. Виклик і завантаження

Формат виконуваного файлу: .exe для Windows. Для запуску треба завантажити виконуваний файл release2.exe

1. Початкові дані

* Характер: прямокутне поле, визначене користувачем.
* Формат введення: цілі числа
* Спосіб кодування: цілочисельний. Значення o означає живу клітину, значення \* — мертву клітину. Дані вводяться та виводяться у вигляді прямокутної таблиці.

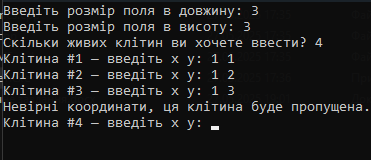


Рисунок 1.1 – введення даних у програмі

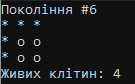


Рисунок 1.2 – виведення даних у програмі

1. Вихідні дані

- Характер: графічне представлення поля у вигляді матриці.

- Формат: консольне виведення з оновленням кожного покоління.

- Спосіб кодування: цілочисельний та символьний. Значення o означає живу клітину, значення \* — мертву клітину. Дані вводяться та виводяться у вигляді прямокутної таблиці.

**ТЕКСТ ПРОГРАМИ**

1. Funcs.h – заголовкий файл функцій

#ifndef FUNCS\_H

#define FUNCS\_H

#include <iostream>

using namespace std;

void fieldcreate(char\*\*&, int&, int&);

void fieldout(char\*\*, int, int);

void fielddelete(char\*\*, int);

void seekAliveCells(char\*\*, int, int);

bool askRestart();

void initConfig(char\*\* f, int w, int h);

int countAliveSusid(char\*\* f, int x, int y, int w, int h);

bool allCellsDead(char\*\* f, int w, int h);

void gameLoop(char\*\*& field, int w, int h);

#endif

1. Andriifuncs.cpp – файл з функціями розроблених Андрієм Жариком

void fieldcreate(char\*\*& f, int& w, int& h) {

cout << "Введіть розмір поля в довжину: ";

cin >> w;

cout << "Введіть розмір поля в висоту: ";

cin >> h;

f = new char\* [h];

for (int i = 0; i < h; i++) {

f[i] = new char[w];

}

for (int i = 0; i < h; i++) {

for (int j = 0; j < w; j++) {

f[i][j] = '\*';

}

}

}

void fieldout(char\*\* f, int w, int h) {

for (int i = 0; i < h; i++) {

for (int j = 0; j < w; j++) {

cout << f[i][j] << " ";

}

cout << endl;

}

}

void fielddelete(char\*\* f, int h) {

for (int i = 0; i < h; i++) {

delete[] f[i];

}

delete[] f;

}

void seekAliveCells(char\*\* f, int w, int h) {

char\*\* temp = new char\* [h];

for (int i = 0; i < h; i++) {

temp[i] = new char[w];

for (int j = 0; j < w; j++) {

temp[i][j] = f[i][j];

}

}

for (int i = 0; i < h; i++) {

for (int j = 0; j < w; j++) {

int count = countAliveSusid(temp, j, i, w, h);

if (temp[i][j] == 'o') {

if (count < 2 || count > 3) f[i][j] = '\*';

}

else {

if (count == 3) f[i][j] = 'o';

}

}

}

for (int i = 0; i < h; i++) {

delete[] temp[i];

}

delete[] temp;

}

1. ihorfuncs.cpp – файл з функціями розроблених Шкуріним Ігорем

#include "funcs.h"

#include <windows.h>

void initConfig(char\*\* f, int w, int h) { //користувач задає кількість живих клітин

int count;

cout << "Скільки живих клітин ви хочете ввести? ";

cin >> count;

for (int i = 0; i < count; i++) { //користувач вводить координати кожної клітини

int x, y;

cout << "Клітина #" << i + 1 << " — введіть x y: ";

cin >> x >> y;

//Перевірка того що координати в межах поля

if (x >= 0 && x < w && y >= 0 && y < h) {

f[y][x] = 'o'; //замінює мертву клітину \* на живу o

}

else {

cout << "Невірні координати, ця клітина буде пропущена." << endl;

}

}

}

int countAliveSusid(char\*\* f, int x, int y, int w, int h) {

int count = 0; // лічильник живих сусідів

//перебирає всі сусідні клітини навколо (x y)

for (int dx = -1; dx <= 1; dx++) {

for (int dy = -1; dy <= 1; dy++) {

if (dx == 0 && dy == 0) continue;

//пропускаємо саму клітину (x y)

int nx = x + dx; //нові координати по x

int ny = y + dy; //нові координати по y

//перевірка, чи не виходять координати за межі поля

if (nx >= 0 && nx < w && ny >= 0 && ny < h) {

//перевіряємо, чи ця сусідня клітина жива

if (f[ny][nx] == 'o') {

count++; //якщо клітина жива - додаємо до лічильника

}

}

}

}

return count;

}

bool allCellsDead(char\*\* f, int w, int h) {

for (int i = 0; i < h; i++) {

for (int j = 0; j < w; j++) {

if (f[i][j] == 'o') {

return false; // знайдена хоча б одна жива клітина

}

}

}

return true; // всі клітини мертві

}

void gameLoop(char\*\*& field, int w, int h) {

int generation = 0;

while (true) {

system("cls"); // очищення екрану

cout << "Покоління #" << generation << endl;

fieldout(field, w, h);

int aliveCount = 0;

for (int i = 0; i < h; i++) {

for (int j = 0; j < w; j++) {

if (field[i][j] == 'o') aliveCount++;

}

}

cout << "Живих клітин: " << aliveCount << endl;

if (aliveCount == 0) {

cout << "Усі клітини мертві. Гру завершено!" << endl;

break;

}

1. main.cpp – основний файл

#include "funcs2.h"

#include <windows.h>

#include <cctype>

bool askRestart() {

char answer;

cout << "Бажаєте почати гру заново? (y/n): ";

cin >> answer;

cin.ignore(10000, '\n'); // очищає залишки вводу

return tolower(answer) == 'y';

}

int main()

SetConsoleOutputCP(65001);

setlocale(LC\_ALL, "RU");

do {

int w, h;

char\*\* field = nullptr;

fieldcreate(field, w, h);

initConfig(field, w, h);

gameLoop(field, w, h);

fielddelete(field, h);

cout << endl;

} while (askRestart());

cout << "Дякуємо за гру!" << endl;

cout << "Натисніть Enter для виходу...";

cin.get();

return 0;

}

**КЕРІВНИЦТВО ПРОГРАМІСТА**

1. Призначення

Програма реалізує класичну гру "Життя" Джона Конвея. Вона дозволяє задати початкову конфігурацію клітин на полі та ітераційно виводить нові покоління відповідно до встановлених правил.

1. Умови застосування програми

* Оперативна пам’ять: не менше 2 Гб
* Архітектура: x86 / x64-сумісні ПК
* Операційна система: Windows XP або новіші
* Необхідні периферійні пристрої: клавіатура, монітор

1. Характеристики програми

* Мова програмування: С++
* Режим роботи: покроковий, з можливістю перегляду кожного покоління
* Тип взаємодії: консольна програма
* Тривалість виконання: залежить від кількості поколінь
* Контроль правильності: перевірка правильності введення координат; програма завершує роботу, якщо немає живих клітин

1. Звертання до програми

Програма запускається шляхом виконання файлу release1.exe. Вся взаємодія відбувається через введення з клавіатури під час виконання.

1. Вхідні дані

* Характер: прямокутне поле, визначене користувачем.
* Формат введення: цілі числа
* Спосіб кодування: цілочисельний. Значення o означає живу клітину, значення \* — мертву клітину. Дані вводяться та виводяться у вигляді прямокутної таблиці.

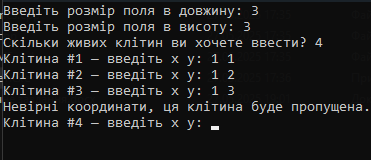


Рисунок 1.3 – введення даних у програмі

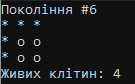


Рисунок 1.4 – виведення даних у програмі

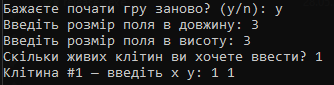


Рисунок 1.4.1– введення даних після підтвердження перезапуску гри

1. Вихідні дані

- Характер: графічне представлення поля у вигляді матриці.

- Формат: консольне виведення з оновленням кожного покоління.

- Спосіб кодування: цілочисельний та символьний. Значення o означає живу клітину, значення \* — мертву клітину. Дані вводяться та виводяться у вигляді прямокутної таблиці

7. Повідомлення

|  |  |  |
| --- | --- | --- |
| Повідомлення | Опис | Дії користувача |
| Введіть розмір поля в довжину: | Запит на розмір гри | Введення числа |
| Введіть розмір поля в висоту: | Запит на розмір гри | Введення числа |
| Скільки живих клітин ви хочете ввести? | Запит кількості живих клітин | Введення числа |
| Усі клітини мертві. Гру завершено! | Кінець гри | Користувач нічого не робить |
| Бажаєте почати гру заново? (y/n): | Запит на перезапуск гри з новою конфігурацією | Користувач вводить y або n |
| Натисніть Enter, щоб завершити... | Запит на натиск Enter | Користувач натискає Enter, програма закривається |

**КЕРІВНИЦТВО СИСТЕМНОГО ПРОГРАМІСТА**

1. Загальні відомості

* Позначення програми: ГЖ.00.01 ПП
* Найменування: «Гра життя»
* Програмне забезпечення: компілятор C++ (MinGW), середовище виконання Windows.
* Мова програмування: C++

1. Опис логічної структури

Алгоритм:

* Користувач задає розміри поля.
* Вводиться початкова конфігурація (живі клітини).
* Підраховується кількість живих сусідів для кожної клітини.
* Оновлюється стан поля згідно з правилами гри:
* Повторюється, поки поле не стане повністю мертвим.
* Запитує, чи не бажає користувач перезапустити гру.

Структура:

* fieldcreate(); - створення початкового поля
* initConfig(); - ініціалізація початкової конфігурації живих клітин користувачем
* gameLoop(); - основний цикл гри
* fielddelete(); - видалення поля
* askRestart(); - запит перезапуску гри

1. Налаштування програми

Перед використанням програма не потребує жодного налаштування, бо немає такого функціоналу, який потребував би налаштування

1. Перевірка програми

Метод контролю:

* Ручне введення координат з відомим результатом
* Візуальний контроль стану поля після кожного покоління

Контрольний приклад:

* "Блінкер" (період осциляції — 2 покоління)
* Розмір поля: 5 x 5
* Початкові координати живих клітин: 1 2, 2 2, 3 2

Результати:

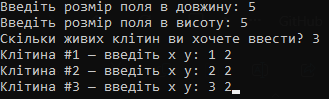


Рисунок 1.5 – ініціалізація початкової конфігурації

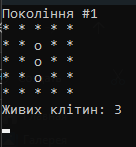


Рисунок 1.6 – перше покоління після заданої конфігурації

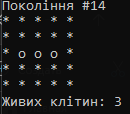


Рисунок 1.7 – друге покоління після заданої конфігурації

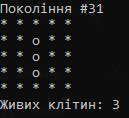
****

Рисунок 1.8 – третє покоління після заданої конфігурації

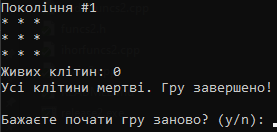


Рисунок 1.9 – запит на перезапуск гри при завершенні

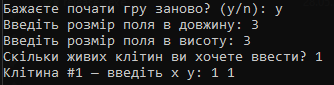


Рисунок 1.10 – введення даних після підтвердження перезапуску гри

1. Додаткові можливості

Під час виконання самостійної роботи було вирішено, що буде спільно написана функція, яка буде перезапускати гру з іншою конфігурацією, після ії завершення

1. Повідомлення системному програмісту

|  |  |  |
| --- | --- | --- |
| Повідомлення | Опис | Дії системного програміста |
| Введіть розмір поля в довжину: | Запит на розмір гри | Введення числа |
| Введіть розмір поля в висоту: | Запит на розмір гри | Введення числа |
| Скільки живих клітин ви хочете ввести? | Запит кількості живих клітин | Введення числа |
| Усі клітини мертві. Гру завершено! | Кінець гри | Системний програміст нічого не робить |
| Бажаєте почати гру заново? (y/n): | Запит на перезапуск гри з новою конфігурацією | Користувач вводить y або n |
| Натисніть Enter, щоб завершити... | Запит на натиск Enter | Системний програміст натискає Enter, програма закривається |

**КЕРІВНИЦТВО ОПЕРАТОРА**

1. Призначення

Програма реалізує класичну гру "Життя" Джона Конвея. Вона дозволяє задати початкову конфігурацію клітин на полі та ітераційно виводить нові покоління відповідно до встановлених правил.

1. Умови застосування програми

* Оперативна пам’ять: не менше 2 Гб
* Архітектура: x86 / x64-сумісні ПК
* Операційна система: Windows XP або новіші
* Необхідні периферійні пристрої: клавіатура, монітор

3. Виконання програми

3.1 Завантаження і запуск

- Запустити виконуваний файл програми.

- Програма запропонує ввести розміри поля.

- Далі ввести кількість живих клітин.

- Після цього по черзі ввести координати кожної живої клітини.

3.2 Керування програмою

- Після початкової ініціалізації програма виведе стан поля..

- Для перезапуску гри вводиться y або n, після ії завершення

- Для завершення роботи — натиснути клавішу Enter

3.3 Формат і можливі варіанти команд

* Введення чисел — лише цілі додатні числа.
* Координати задаються у форматі: рядок, стовпець
* Команди керування — клавіші Enter (завершення).

4.Повідомлення оператору

|  |  |  |
| --- | --- | --- |
| Повідомлення | Опис | Дії оператора |
| Введіть розмір поля в довжину: | Запит на розмір гри | Введення числа |
| Введіть розмір поля в висоту: | Запит на розмір гри | Введення числа |
| Скільки живих клітин ви хочете ввести? | Запит кількості живих клітин | Введення числа |
| Усі клітини мертві. Гру завершено! | Кінець гри | Оператор нічого не робить |
| Бажаєте почати гру заново? (y/n): | Запит на перезапуск гри з новою конфігурацією | Користувач вводить y або n |
| Натисніть Enter, щоб завершити... | Запит на натиск Enter | Оператор натискає Enter, програма закривається |