

Міністерство освіти і науки України
Національний університет «Запорізька Політехніка»

кафедра програмних засобів

ЗВІТ

з розрахунково-графічного завдання № 2

з дисципліни “Soft skills, групова динаміка та комунікації” на тему:
«КОМАНДНА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

Виконали:

студентки групи КНТ-132

А.Р. Лещинська

М.Г. Кочева

Прийняв:

доцент

В.М. Льовкін

2023

1 КОМАНДНА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Мета роботи

1.1.1 Навчитися розробляти програмне забезпечення, працюючи в команді.

1.1.2 Навчитися оформлювати програмну документацію.

1.2 Завдання до роботи

1.2.1 Ознайомитися з основними теоретичними відомостями за темою роботи, використовуючи дані методичні вказівки, а також рекомендовану літературу.

1.2.2 Сформувати команду з двох студентів та отримати у викладача індивідуальне завдання.

1.2.3 Завести акаунт на сервісі GitHub та створити команду для роботи в Slack.

1.2.4 Узгодити розподіл зобов'язань для виконання завдання, використовуючи Slack.

1.2.5 Створити репозиторій для роботи над проектом та дозволити доступ до проекту обом користувачам.

1.2.6 Завантажити у віддалений репозиторій існуючі файли проекту.

1.2.7 Створити дві гілки проекту: для презентації ревізій та для відлагодження проекту.

1.2.8 Налаштувати доступ до Git-репозиторію в інтегрованому середовищі розробки Eclipse.

1.2.9 Виконати реалізацію проекту в Eclipse та зазначити, яка частина коду ким була розроблена.

1.2.10 Визначити додаткову функцію та реалізувати її, працюючи над нею одночасно вдвох.

1.2.11 Завершити роботу над проектом.

1.2.12 Відповідно до діючих стандартів індивідуально оформити програмний документ, узгоджений з викладачем, на розроблене програмне забезпечення.

1.2.13 Оформити звіт з роботи.

1.2.14 Відповісти на контрольні запитання.

1.3 Спільне завдання

Гра відбувається на прямокутному полі. Розмір поля вибирається гравцем. Поле складається з клітин. У кожної клітини є 8 сусідів. Правила такі:

- жива клітина, у якій є менше ніж дві живі клітини серед сусідів, вмирає;
- жива клітина, у якій більш ніж три живі клітини серед сусідів, також вмирає;
- жива клітина, у якій дві або три живі клітини серед сусідів, продовжує жити;
- мертва клітина, у якій рівно три живі клітини серед сусідів, стає живою.

Початкова конфігурація задається гравцем. Гра закінчується тоді, коли на полі немає більше живих клітин, інакше розподіл клітин повторюється по кроках.

На кожному кроці відображається поле, що складається з клітин. Кожна клітина відображається цифрою або кольором (якщо готові зробити графічне представлення), що відповідає стану клітини. Потрібно послідовно показати всі кроки.

1.4 Основні теоретичні відомості

Для роботи з Git в інтегрованих середовищах розробки існують спеціальні плагіни, наприклад, EGit для Eclipse.

Для того щоб інсталиувати даний плагін, необхідно в Eclipse обрати пункт меню Help → Eclipse Marketplace та в рядку пошуку ввести EGit, після чого інсталиувати відповідний плагін. Далі необхідно створити репозиторій на основі нового проекту. Після того, як проект створено, необхідно обрати з контекстного меню проекту Team → Share Project та виконати конфігурацію репозиторію. У результаті проект належить репозиторію на основі гілки master. Для відслідковування змін необхідно з контекстного меню обрати Team → Add to Index. У результаті файли проекту додано до контролю версій. Далі можна виконати фіксацію, викликавши пункт меню Team → Commit.

Для налаштування SSH в Eclipse (рис. 2.4) при підключенні до GitHub необхідно обрати пункт меню Window → Preferences і далі General → Network Connections → SSH2. У даному вікні необхідно перевірити наявність ключів та коректність налаштування.

Основними документами при розробленні програмного забезпечення є:

- технічне завдання;
- специфікація;
- опис програми;
- текст програми;
- керівництво програміста;
- керівництво системного програміста;
- керівництво оператора.

Згідно з ГОСТом 19.201–78 технічне завдання повинно містити такі розділи:

- 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.
- вимоги до транспортування та збереження;
- спеціальні вимоги.

У підрозділі «Умови експлуатації» повинно бути зазначено умови експлуатації (температура навколишнього повітря, відносна вологість тощо для обраних типів носіїв даних), при яких повинні забезпе.

Згідно з ГОСТом 19.202-78 специфікація повинна містити розділи:

- документація;
- комплекси;
- компоненти.

Специфікацію оформлюють у вигляді таблиці. Найменування кожного розділу вказують у виді заголовка в графі «Найменування». Для документів, виконаних друкованим способом, заголовки підкреслюють.

У розділ «Документація» вносять програмні документи на дану програму, крім специфікації і технічного завдання, у порядку зростання коду виду документа, що входить у позначення.

Далі записують запозичені програмні документи. Запис їх виконується в порядку зростання кодів підприємств-розробників і далі в порядку зростання коду виду документа, що входить у позначення.

Після кожного розділу специфікації необхідно залишати декілька вільних рядків для додаткових записів.

Згідно з ГОСТом 19.402–78 опис програми повинен містити на ступні розділи:

- загальні відомості;
- функціональне призначення;
- опис логічної структури;
- використані технічні засоби;
- виклик і завантаження;
- початкові дані, вихідні дані.

У залежності від особливостей програми допускається вводити додаткові розділи чи поєднувати окремі розділи.

У розділі «Загальні відомості» повинно бути зазначено: – позначення та найменування програми;

- програмне забезпечення, необхідне для функціонування програми;
- мови програмування, якими написана програма.

У розділі «Функціональне призначення» повинно бути зазначено класи розв’язуваних задач і (або) призначення програми та відомості про функціональні обмеження на застосування.

У розділі «Опис логічної структури» повинно бути зазначено: алгоритм програми, використані методи, структура програми з описом функцій складових частин і зв’язку між ними, зв’язку програми з іншими програмами.

Опис логічної структури програми виконують з урахуванням тексту програми вихідною мовою.

Згідно з ГОСТом 19.504–79, керівництво програміста має містити наступні розділи:

- призначення та умови застосування програми;
- характеристики програми;
- звернення до програми;
- вхідні та вихідні дані;

– повідомлення.

У залежності від особливостей документа допускається поєднувати окремі розділи або вводити нові.

У розділі «Призначення та умови застосування програми» повинно бути зазначено призначення та функції, які виконуються програмою, умови, необхідні для виконання програми (обсяг оперативної пам'яті, вимоги до складу та параметрів периферійних пристроїв, вимоги до програмного забезпечення тощо).

У розділі «Характеристика програми» повинно бути приведено опис основних характеристик і особливостей програми (часові характеристики, режим роботи, засоби контролю правильності виконання та самовідновлення програми тощо).

У розділі «Звертання до програми» повинно бути наведено опис процедур виклику програми (способи передачі управління та параметрів даних тощо).

У розділі «Початкові та вихідні дані» повинно бути приведено опис організації початкової і вихідної інформації, що використовується, та за необхідності її кодування.

У розділі «Повідомлення» повинно бути зазначено тексти повідомлень, які видаються програмісту чи оператору в процесі виконання програми, опис їхнього змісту та дії, які необхідно виконати за появи цих повідомлень.

У додатку до керівництва програміста можуть бути наведені додаткові матеріали (приклади, ілюстрації, таблиці, графіки тощо).

1.5 Код програми

```
#include <iostream>
```

```
using namespace std;
```

```

int main(){

    int n, m;
    cout << "N = ";
    cin >> n;
    cout << "M = ";
    cin >> m;

    cout << endl << "Enter cell states(1 if alive, 0 if dead, if you enter any other
number - it will be count as dead):\n";
    n += 2;
    m += 2;

    int a[n][m];
    for(int i = 0, j = n - 1, s = 0; s < m; s++){
        a[i][s] = 0;
        a[j][s] = 0;
    }

    for(int i = 0, j = m - 1, s = 0; s < n; s++){
        a[s][i] = 0;
        a[s][j] = 0;
    }

    for(int i = 1; i < n - 1; i++)
        for(int j = 1; j < m - 1; j++){
            cin >> a[i][j];
            if(a[i][j] != 1 && a[i][j] != 0) a[i][j] = 0;
        }
}

```



```

cout << "\nGame starts:\n";
for(int i = 0; i < n; i++, cout << endl)
    for(int j = 0; j < m; j++) cout << a[i][j] << " ";

cout << endl;

while(true){
    int c = 0;
    for(int i = 1; i < n - 1; i++)
        for(int j = 1; j < m - 1; j++){
            int an = 0;
            if(a[i-1][j] == 1) an++;
            if(a[i+1][j] == 1) an++;
            if(a[i][j-1] == 1) an++;
            if(a[i][j+1] == 1) an++;
            if(a[i-1][j-1] == 1) an++;
            if(a[i-1][j+1] == 1) an++;
            if(a[i+1][j-1] == 1) an++;
            if(a[i+1][j+1] == 1) an++;
            if(an != 2 && an != 3 && a[i][j] == 1){
                c++;
                a[i][j] = 0;
            }
            else if((an == 2 || an == 3) && a[i][j] == 0){
                c++;
                a[i][j] = 1;
            }
        }
    if(c == 0){
        cout << "Game ended!" << endl;
    }
}

```

```

    break;
}
for(int i = 0; i < n; i++, cout << endl)
    for(int j = 0; j < m; j++) cout << a[i][j] << " ";
    cout << endl;
}
return 0;
}

```

1.6 Опис розподілу відповідальності в команді

За Анною стояло завдання зробити першу частину коду:

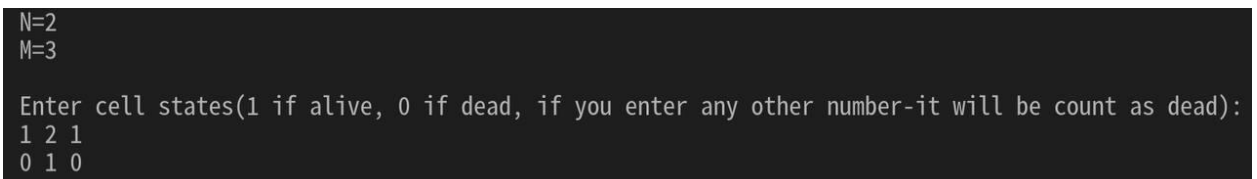
- запит розмірності ігрового поля;
- введення статусу життєздатності для відповідних клітин;
- виведення всього ігрового поля.

В той час, як Мілена прописала:

- алгоритм підрахунку кількості живих сусідів-клітин для поточної клітини;
- обробку ситуацій з відмиранням або оживанням клітин, котрі задовільняють умові завдання.

Потім разом, в режимі онлайн у додатку Slack з'єднували дві частини коду. Далі, після фінального налагодження коду розпочали звіт. Анна робила керівництво програміста, Мілена – керівництво оператора.

1.7 Скріншоти



```

N=2
M=3

Enter cell states(1 if alive, 0 if dead, if you enter any other number-it will be count as dead):
1 2 1
0 1 0

```

Рисунок 1.1 – Введення розмірності та заповнення поля користувачем

```

Game starts:
0 0 0 0 0
0 1 0 1 0
0 0 1 0 0
0 0 0 0 0

0 0 0 0 0
0 0 1 1 0
0 1 1 1 0
0 0 0 0 0

0 0 0 0 0
0 1 0 1 0
0 1 0 0 0
0 0 0 0 0

0 0 0 0 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 0

0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0

Game ended!

```

Рисунок 1.2 – Просимульована гра

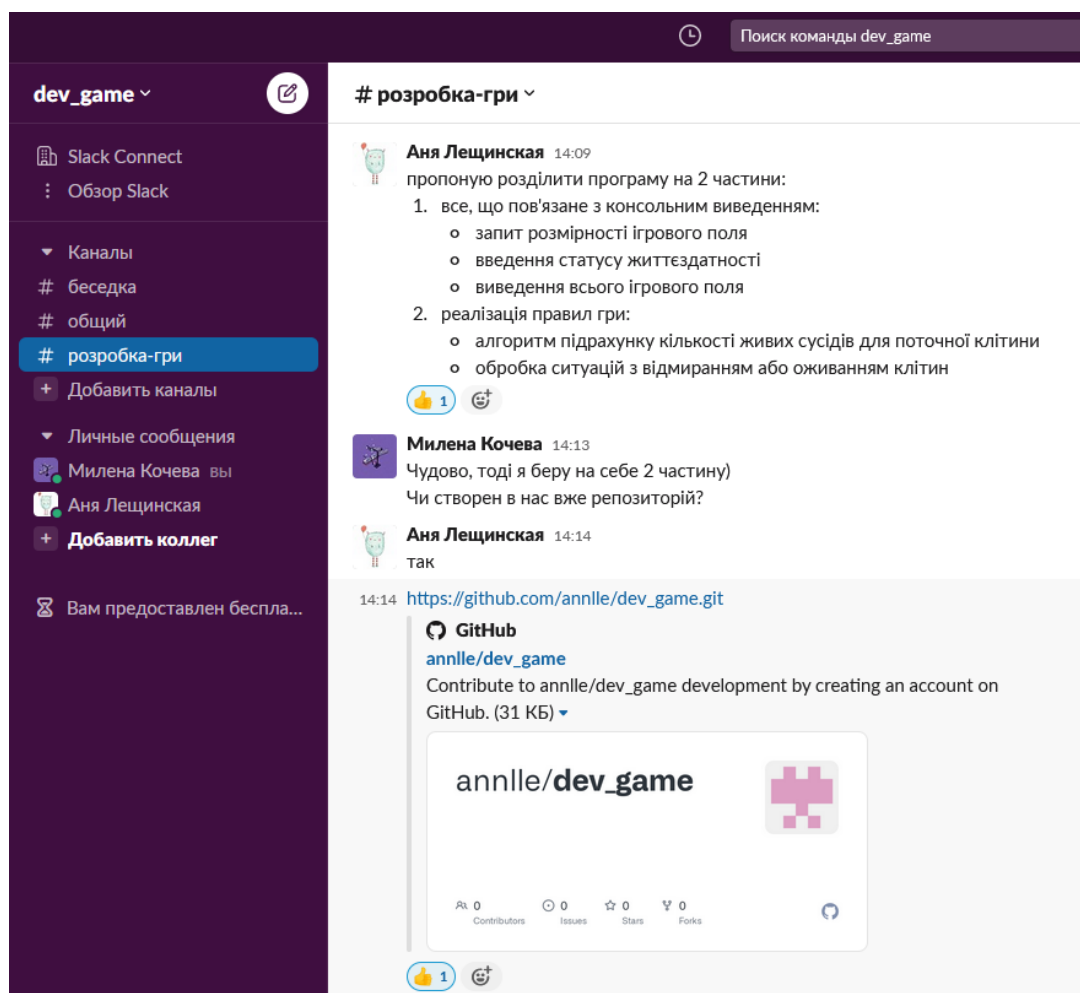


Рисунок 1.3 – Створені акаунти та узгодження ролей у Slack



Аня Лещинская 17:10

я свою часть доробила

```
#include<iostream>
using namespace std;
int main(){
    int n,m;
    cout<<"N=";
    cin>>n;
    cout<<"M=";
    cin>>m;
    cout<<endl<<"Enter cell states(1 if alive, 0 if dead, if you enter any other number-it will be count as
dead):\n";
    n+=2;
    m+=2;
    int a[n][m];
    for(int i=0,j=n-1,s=0;s<m;s++){
        a[i][s]=0;
        a[j][s]=0;
    }
    for(int i=0,j=m-1,s=0;s<n;s++){
        a[s][i]=0;
        a[s][j]=0;
    }
    for(int i=1;i<n-1;i++){
        for(int j=1;j<m-1;j++){
            cin>>a[i][j];
            if(a[i][j]!=1 && a[i][j]!=0)
                a[i][j]=0;
        }
    }
    cout<<"\nGame starts:\n";
    for(int i=0;i<n;i++,cout<<endl)
        for(int j=0;j<m;j++)
            cout<<a[i][j]<<" ";
            cout<<endl;
    for(int i=0;i<n;i++,cout<<endl)
        for(int j=0;j<m;j++)
            cout<<a[i][j]<<" ";
            cout<<endl;
    }
    return 0;
}
```

Рисунок 1.4 – Часть кода Анни




Милена Кочева 17:24


Добре, додаю реалізацію самого принципу гри:

```
while(true){
    int c = 0;
    for(int i = 1; i < n - 1; i++){
        for(int j = 1; j < m - 1; j++){
            int an = 0;
            if(a[i-1][j] == 1) an++;
            if(a[i+1][j] == 1) an++;
            if(a[i][j-1] == 1) an++;
            if(a[i][j+1] == 1) an++;
            if(a[i-1][j-1] == 1) an++;
            if(a[i-1][j+1] == 1) an++;
            if(a[i+1][j-1] == 1) an++;
            if(a[i+1][j+1] == 1) an++;
            if(an != 2 && an != 3 && a[i][j] == 1){
                c++;
                a[i][j] = 0;
            }
            else if((an == 2 || an == 3) && a[i][j] == 0){
                c++;
                a[i][j] = 1;
            }
        }
    }
    if(c == 0){
        cout << "Game ended!" << endl;
        break;
    }
    for(int i = 0; i < n; i++, cout << endl)
        for(int j = 0; j < m; j++) cout << a[i][j] << " ";
    cout << endl;
}
```

Рисунок 1.5 – Частина коду Мілени


 **dev_game** Public


Watch 1 Fork 0 Star 0


 **debug** had recent pushes 13 minutes ago Compare & pull request

main Go to file Add file Code

Branches Tags

 **annlle** Init commit 3 hours ago 1

 **.gitignore** Init commit 3 hours ago

 **game.cpp** Init commit 3 hours ago

Help people interested in this repository understand your project by adding a README. Add a README

Releases
No releases published
[Create a new release](#)

Рисунок 1.4 – Репозиторій для роботи з проектом

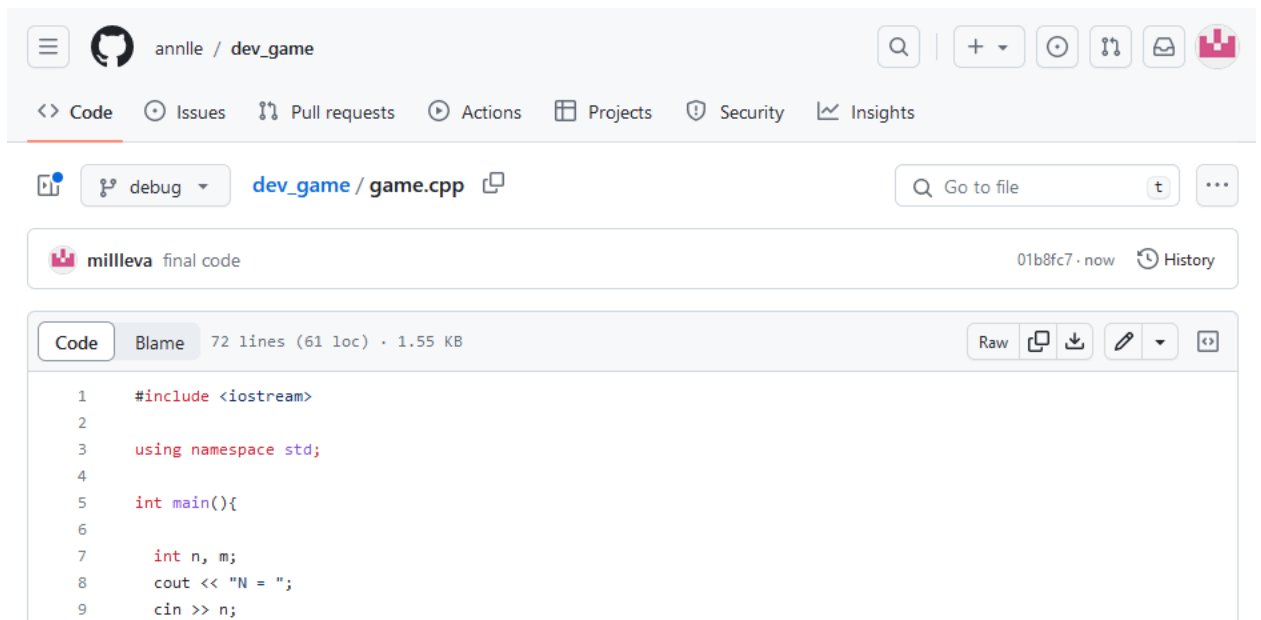


Рисунок 1.6 – Завантажили у віддалений репозиторій код

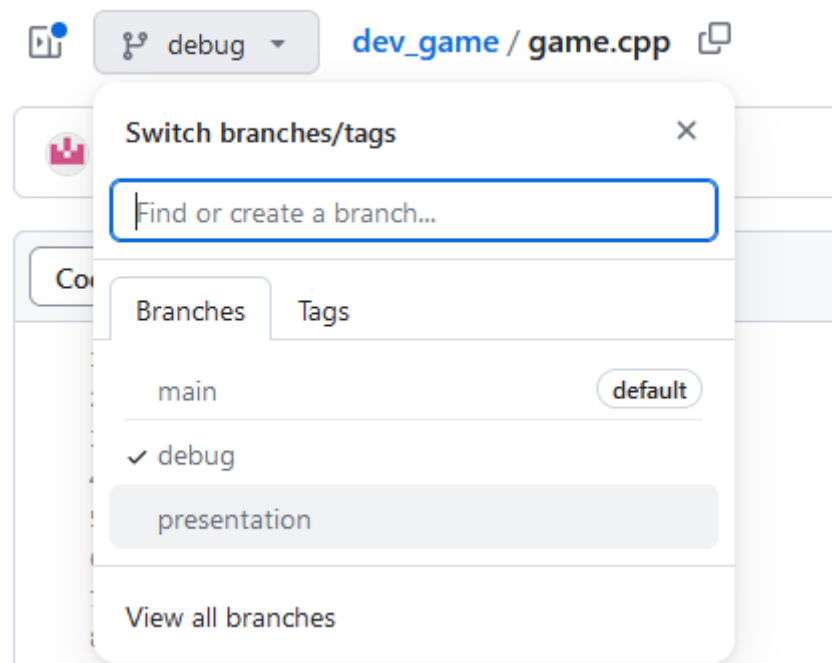


Рисунок 1.7 – Дві гілки проекту: для презентації ревізій та для відлагодження проекту

1.8 Висновки:

В цьому проекті ми навчилися розробляти програмне забезпечення, працюючи в команді. Також навчилися оформлювати програмну документацію.

1.8.1 Обґрунтуйте необхідність оформлення програмної документації.

Оформлення програмної документації є необхідним для ефективного розроблення, управління та підтримки програмного продукту. Вона надає структуровану та зрозумілу інформацію про функції, архітектуру, логіку роботи та інші аспекти програми, що допомагає розробникам, тестувальникам і замовникам у розумінні проекту.

1.8.2 Відповідно до якого стандарту оформляється документ «Технічне завдання»?

Згідно з ГОСТом 19.201–78

1.8.3 Згідно з яким стандартом оформляється документ «Специфікація»?

Згідно з ГОСТом 19.202-78

1.8.4 З яких розділів складається специфікація?

- документація;
- комплекси;
- компоненти.

1.8.5 З яких розділів складається керівництво системного програміста?

- загальні відомості про програму;
- структура програми;
- налаштування програми;
- перевірка програми;
- додаткові можливості;
- повідомлення системному програмісту.

1.8.1 Які основні документи оформлюються при розробленні програмного забезпечення?

- технічне завдання;
- специфікація;
- опис програми;
- текст програми;
- керівництво програміста;
- керівництво системного програміста;
- керівництво оператора.

1.8.2 З написання якого документу починається процес розроблення програмного забезпечення?

З технічного завдання.

1.8.3 Відповідно до якого стандарту оформляється документ «Технічне завдання»?

Відповідно до ГОСТу 19.201-78.

1.8.4 Відповідно до якого стандарту оформляється документ «Текст програми»?

Відповідно до ГОСТу 19.401-78.

1.8.5 Які розділи містить керівництво оператора?

- призначення програми;
- умови виконання програми;
- виконання програми;
- повідомлення оператору.