POLITECHNIKA
RZESZOWSKA
im. IGNACEGO ŁUKASIEWICZA

Wydział Matematyki i Fizyki Stosowanej

Inżynieria i analiza danych

PROJEKT

Usługi sieciowe w biznesie

Gra kółko i krzyżyk - GIT

Daniel Krzysik 166667

Rzeszów 2022

Spis treści

1. Wstęp

W dzisiejszych czasach programowanie stało się bardzo popularne, a wraz z nim także korzystanie z różnych narzędzi, takich jak git czy Github. Dlatego też postanowiłem nauczyć się tych narzędzi oraz wykorzystać je w swoim projekcie, którym jest gra kółko i krzyżyk w języku R.

Celem pracy jest stworzenie funkcjonalnej i poprawnie działającej gry, która umożliwi użytkownikom granie w kółko i krzyżyk na planszy 3x3. W ramach pracy zostanie omówiona koncepcja projektu oraz zastosowane metody, które pozwoliły na jego realizację. Opiszę także korzystanie z systemu kontroli wersji git oraz umieszczenie projektu na platformie Github.

Wierzę, że ta praca pozwoli mi nie tylko na rozwijanie swoich umiejętności programistycznych, ale także na poznanie nowych narzędzi, które pozwolą mi na jeszcze skuteczniejszą i efektywniejszą pracę.

2. Git

Co to jest Git i dlaczego cieszy się tak dużą popularnością? Ten system kontroli wersji znacznie usprawnia, a jednocześnie zabezpiecza codzienną pracę przy kodzie. Dzięki swojej prostocie i elastyczności może być wykorzystywany zarówno przy drobnych, jak i ogromnych projektach. Dlatego też jest używany przez programistów oraz grafików na całym świecie.

Odpowiadając w skrócie na pytanie, co to jest Git, należy powiedzieć, że to system kontroli wersji. Służy on więc do zarządzania historią kodu źródłowego. Jego funkcjonalność ma kilka podłoży. Między innymi sprawdza się tak dobrze, ponieważ:

- pozwala na jednoczesną pracę na tym samym kodzie przez kilka osób,
- umożliwia transferowanie i łączenie zmian z różnych branchy w jednym projekcie,
- pozwala na pracę offline we własnym repozytorium,
- jest szybki i wydajny.

Cechy te sprawiły, że Git szybko został doceniony w całej branży. Przechowywanie wersji, a także możliwość rozgałęziania kodu to niewątpliwie jego ogromne zalety.

2.1 Git – podstawowe funkcje

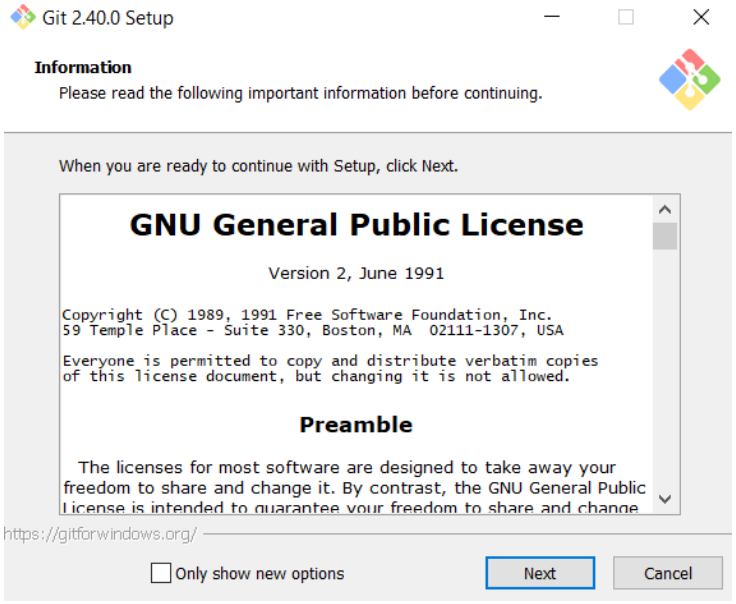
Do najważniejszych pojęć należą:

- SCM – Source Code Management, czyli po prostu kontrola wersji, a więc to, co zawdzięczamy systemowi Git;
- repozytorium – miejsce przechowywania całego projektu, wszystkich wersji kodu i historii wprowadzonych zmian;
- branch – pojedyncze odgałęzienie, jedna wersja, na której pracuje dany programista;
- clone – pozwala na skopiowanie kodu z repozytorium do własnej gałęzi;
- commit – przesyła dane z Twojej gałęzi do repozytorium;
- merge – łączy zmiany wprowadzone w poszczególnych branchach.

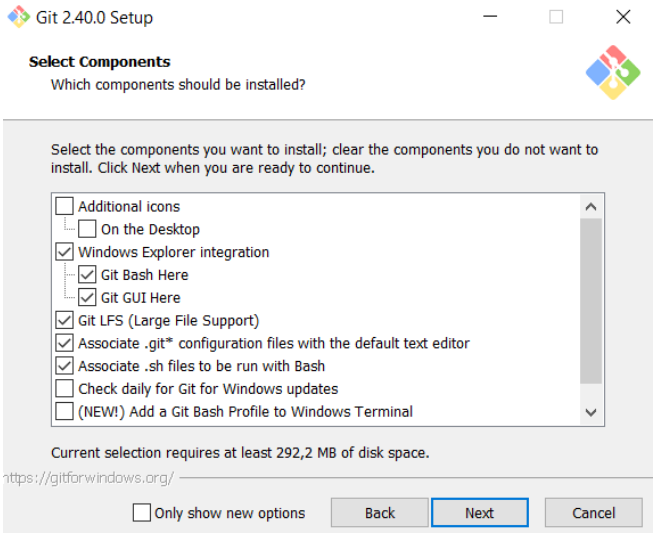
2.2 Instalacja Gita

1. Pobierz instalator dla systemu Windows ze strony https://git-scm.com/download/win
2. Uruchom pobrany instalator

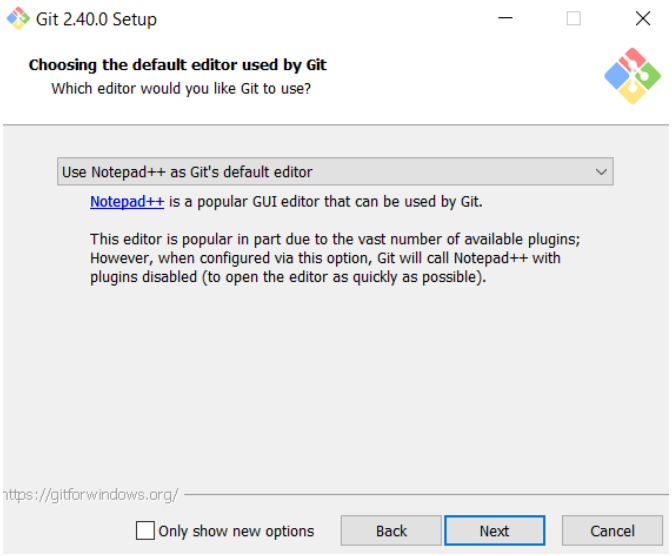
3. Zaakceptuj warunki licencji i kliknij przycisk "Next".



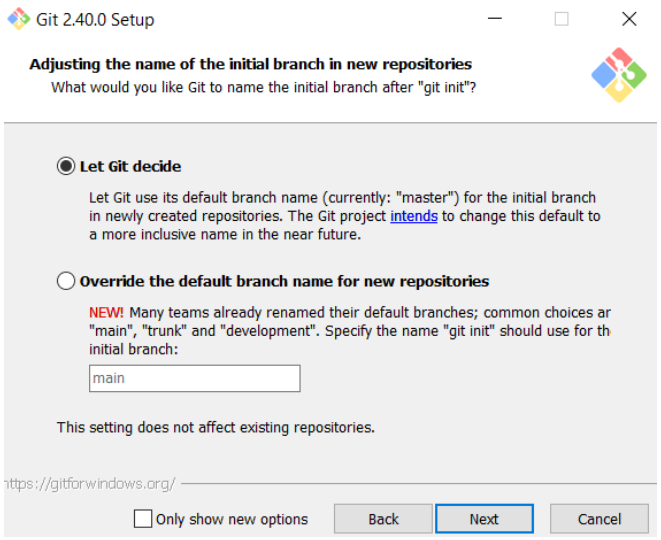
4. Wybierz składniki, które chcesz zainstalować. Domyślnie zaznaczona jest opcja "Git Bash Here", która pozwala na uruchomienie konsoli Git Bash z poziomu menu kontekstowego w Eksploratorze plików.



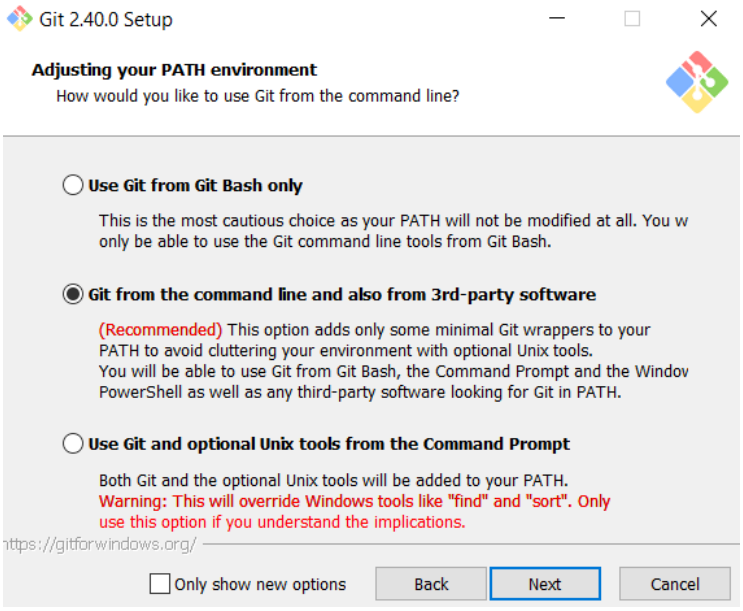
5. Wybierz edytor tekstu, który chcesz używać z Gitem. Domyślnie wybrany jest edytor Nano, ale możesz wybrać dowolny inny edytor, np. Notepad++ . Kliknij przycisk "Next".



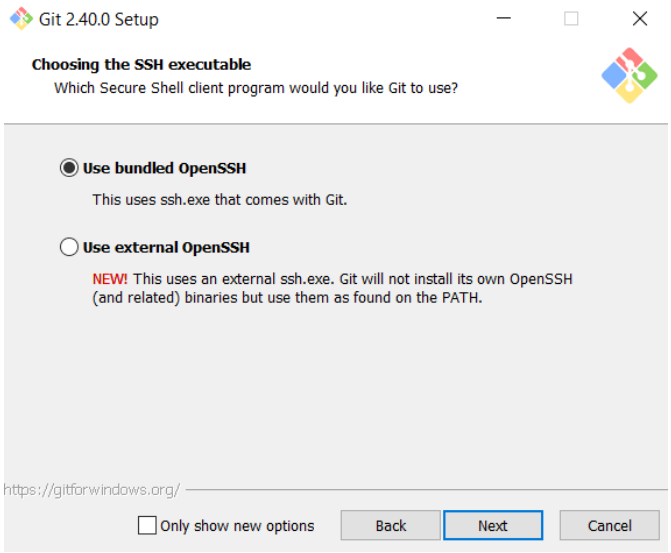
6. W tym oknie można wybrać, jak nazwę ma mieć domyślny branch w nowo tworzonych repozytoriach. Domyślnie Git proponuje nazwę "master", ale możesz wybrać również niestandardową nazwę. Jeśli wybierzesz niestandardową nazwę, pojawi się pole tekstowe, w którym możesz wpisać nazwę swojego wyboru.



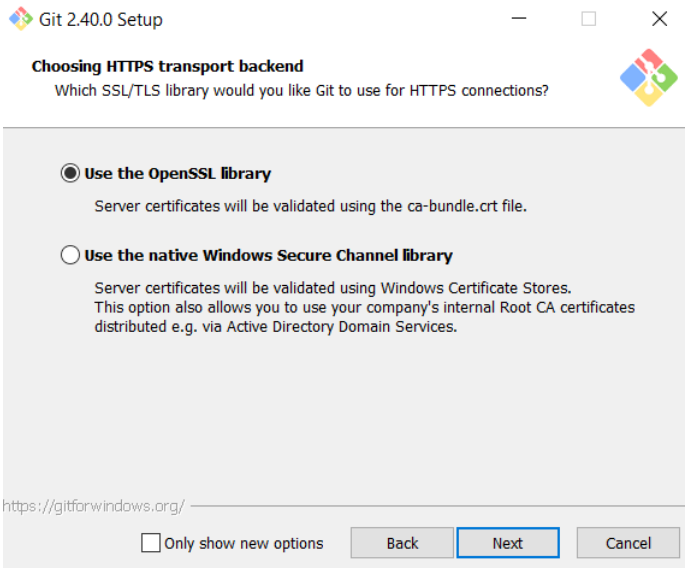
7. Poniżej decydujemy jak mocno zintegrowany z Windowsem ma być nasz git



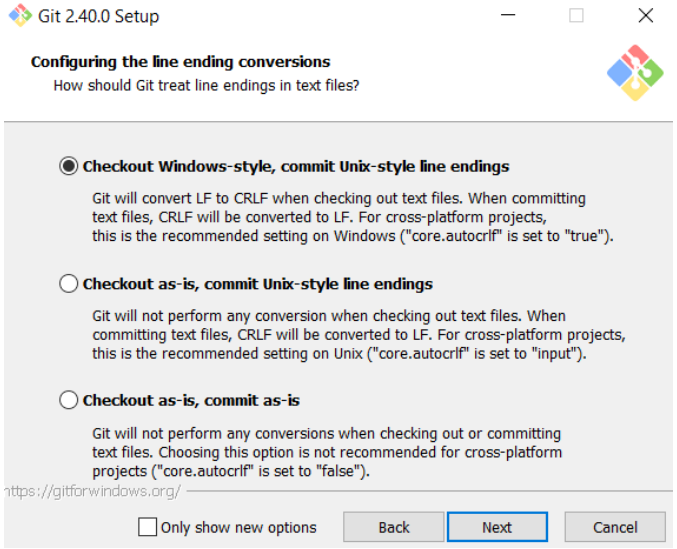
8. Poniższy krok instalacji Gita dotyczy wyboru klienta SSH, który będzie używany przez Gita podczas połączeń z zdalnymi serwerami.



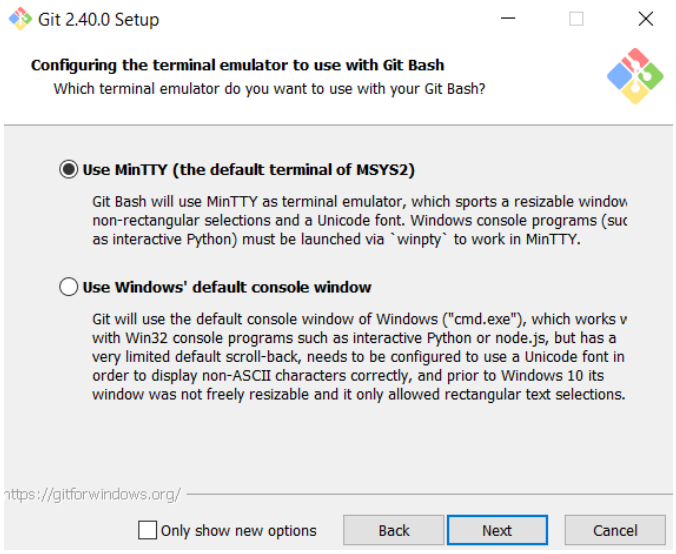
9. Ten krok instalacji Git'a dotyczy wyboru sposobu, w jaki Git będzie korzystał z protokołu HTTPS podczas łączenia się z zdalnym repozytorium.



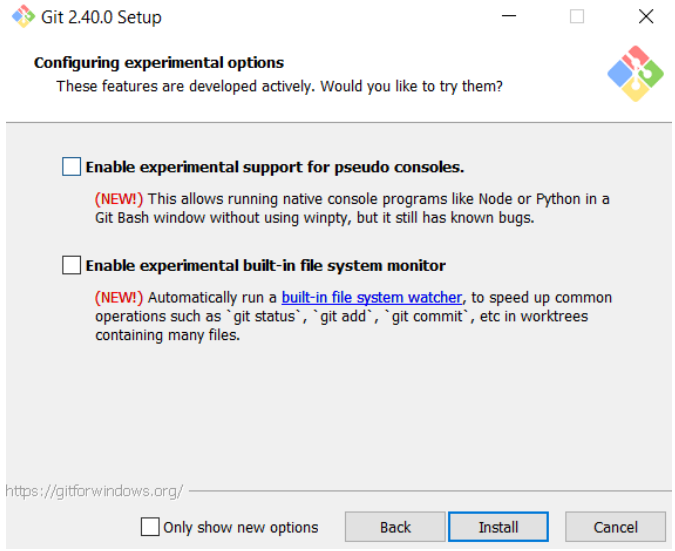
10. Konfiguracja konwersji końców linii.



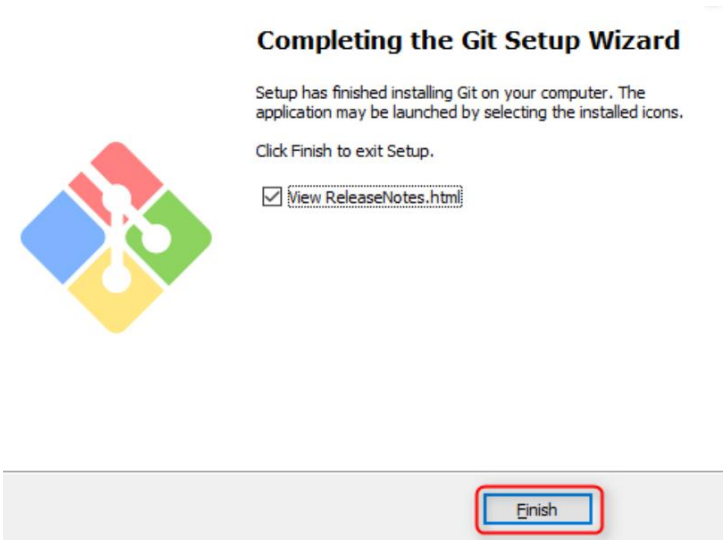
11. Krok "configurating the terminal emulator to use with git bash" podczas instalacji Git'a dotyczy wyboru preferowanego terminala do użytku z Git Bash.



12. Krok "configuring experimental options" podczas instalacji Git'a umożliwia skonfigurowanie eksperymentalnych opcji dla Gita. Te opcje są zazwyczaj dostępne tylko dla zaawansowanych użytkowników i nie są zalecane dla początkujących. Kliknij przycisk "Install", aby rozpocząć instalację.



13. Po zakończeniu instalacji kliknij przycisk "Finish", aby zakończyć proces instalacji.



2.3 Konfiguracja gita

Na początku musimy wstępnie skonfigurować naszą instalację Gita. Najważniejsze to ustawienie nazwy użytkownika i adresu e-mail. Dokonuje się tego wpisując poniższe polecenia:

git config –global user.name nasz_login

```
$ git config --global user.name krzysikd
```

git config –global user.mail nasz_main@gmail.com

```
$ git config --global user.email danielkrzysik00@gmail.com
```

3 Github

GitHub to usługa hostingu umożliwiająca zarządzanie repozytoriami Git. Przy jego pomocy jesteśmy w stanie udostępnić swój kod w jednym miejscu dla wszystkich. Dzięki temu – w tym samym czasie – zapewniona jest możliwość aktywnej współpracy z pozostałymi członkami projektu. GitHub w przeciwieństwie do samego Gita działa w oparciu o chmurę. Każdy członek projektu może bez względu na szerokość geograficzną i sprzęt, na którym działa uzyskać zdalny dostęp do repozytorium Git (warunkiem jest dostęp do sieci).

4. Środowisko programistyczne R

Program R jest projektem GNU opartym o licencje GNU GPL, oznacza to, iż jest w zupełności darmowy zarówno do zastosowań edukacyjnych jak i biznesowych. Od początku język R był tworzony i rozwijany pod statystyków, z tego też powodu przez informatyków często nie był traktowany jak pełnowartościowy język, ale jak język domenowy (DSL, ang. Domain Specific Language). Z czasem jednak możliwości R rosły, pojawiło się coraz więcej rozwiązań wykraczających poza analizę danych i dziś R jest jednych z popularniejszych języków programowania.

4.1 Instalacja R

Instalacja jest prosta, wystarczy otworzyć stronę https://cran.r-project.org/, wybrać system operacyjny i pobrać plik binarny do instalacji. Instalacja sprowadza się do klikania przycisku Next, next, next. Można mieć zainstalowanych jednocześnie kilka różnych wersji R. Przechowywanie starszych wersji programu R może być wygodne, jeżeli chcemy mieć prostą możliwość odtworzenia w przyszłości dokładnie tych samych wyników co na wersji obecnej.

4.2 Instalacja Rstudio

RStudio jest narzędziem ułatwiającym pracę z R. Jest to edytor, manager wersji, narzędzie wspierające debugowanie, tworzenie pakietów, aplikacji czy raportów. Można żyć bez tej nakładki, ale co to za życie.

Najnowszą wersję RStudio Desktop można pobrać ze strony:

http://www.rstudio.com/products/rstudio/download/.

Nie musimy pamiętać tego adresu, wystarczy wpisać w Google R Studio download a powyższy adres będzie pierwszym linkiem.

Praca z RStudio jest znacznie przyjemniejsza gdy nauczymy się kilku podstawowych skrótów klawiszowych. Pełną listę skrótów można znaleźć pod adresem: https://www.rstudio.com/wp-content/uploads/2016/01/rstudio-IDE-cheatsheet.pdf

5. Tworzenie gry kółko i krzyżyk w języku R

5.1 Analiza problemu

Gra kółko i krzyżyk polega na uzupełnianiu planszy o wymiarach 3x3 symbolami "x" i "o". Gracze na przemian stawiają swój symbol na wolnym polu. Wygrywa ten, kto jako pierwszy ustawi trzy swoje symbole w rzędzie, kolumnie lub na przekątnej. W przypadku, gdy wszystkie pola zostaną wypełnione, a żaden z graczy nie osiągnie celu, gra kończy się remisem.

5.2 Projektowanie struktury gry

Do reprezentacji planszy gry użyto wektora stanu (stan_poczatkowy) o długości dziewięć elementów. Elementy tego wektora odpowiadają polom planszy i mogą mieć wartości "x", "o" lub liczby od 1 do 9, gdy pole jest wolne.

Do przechowywania możliwych kombinacji wygrywających użyto listy (trojka). Lista składa się z ośmiu elementów, każdy zawierający wektor o długości trzy, reprezentujący indeksy wygrywającej kombinacji na planszy.

W projekcie zaimplementowano następujące funkcje:

- wyswietl(stan) - wyświetla aktualny stan planszy
- aktualizacja(stan, kto, pozycja) - aktualizuje stan planszy po ruchu gracza
- zwyciezca(stan) - sprawdza, czy któryś z graczy wygrał
- tura_komputera(stan) - realizuje ruch komputera
- rozpocznij_gre() - rozpoczyna grę i steruje jej przebiegiem

5.3 Implementacja funkcji gry

- Funkcja wyswietl

Funkcja wyswietl() przyjmuje jako argument aktualny stan planszy i wyświetla go w czytelnej formie. Wykorzystuje funkcję cat() oraz sprintf() do formatowania tekstu.

- Funkcja aktualizacja

Funkcja aktualizacja() przyjmuje trzy argumenty: stan planszy, symbol gracza oraz pozycję, na której gracz chce postawić swój symbol. Funkcja zwraca nowy stan planszy z uwzględnieniem ruchu gracza.

- Funkcja zwyciezca

Funkcja zwyciezca() przyjmuje jako argument stan planszy. Sprawdza, czy któryś z graczy wygrał, porównując aktualny stan planszy z wcześniej zdefiniowanymi wygrywającymi kombinacjami (trojka). Zwraca wartość logiczną TRUE, jeśli jeden z graczy wygrał, lub FALSE, gdy nikt nie wygrał.

- Funkcja tura_komputera

Funkcja tura_komputera() przyjmuje jako argument stan planszy. Realizuje ruch komputera, wybierając odpowiednie pole do postawienia swojego symbolu. Funkcja korzysta z podejścia heurystycznego, polegającego na sprawdzeniu, czy komputer może wygrać lub zablokować przeciwnika. Jeśli takie ruchy nie są możliwe, komputer wybiera losowe wolne pole. Funkcja zwraca nowy stan planszy po ruchu komputera.

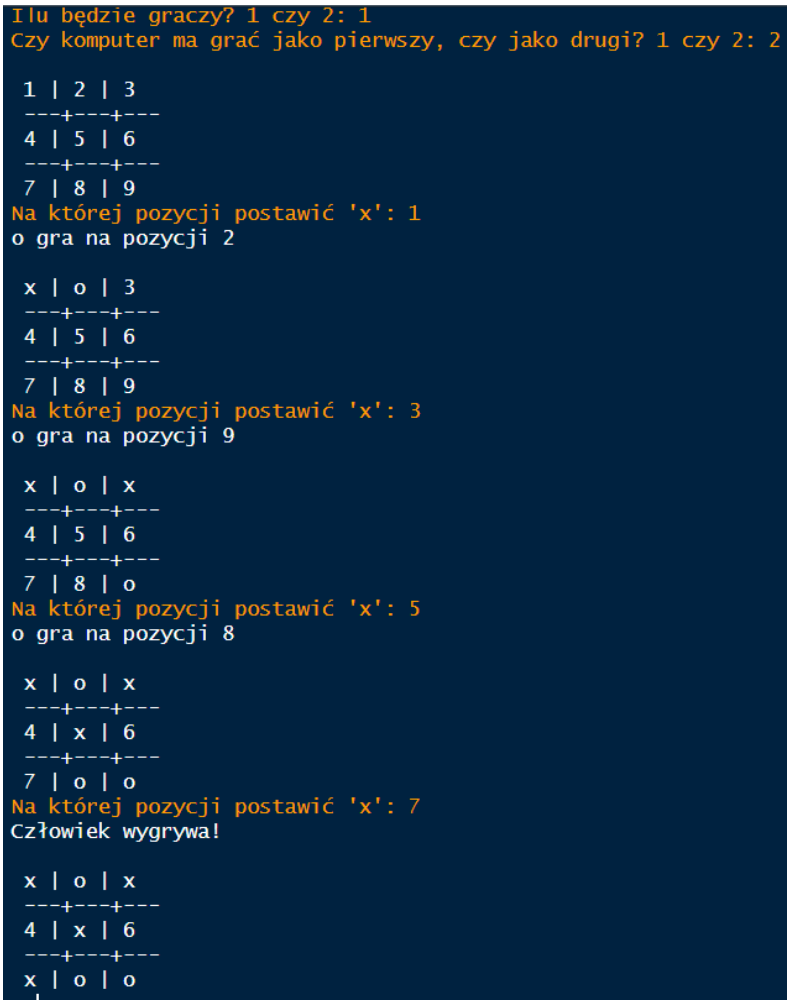
- Funkcja rozpocznij_gre

Funkcja rozpocznij_gre() inicjuje grę, pytając użytkownika o liczbę graczy oraz kolejność ruchów. Następnie realizuje rozgrywkę, na przemian wykonując ruchy graczy i komputera, wyświetlając planszę oraz sprawdzając, czy któryś z graczy wygrał. Gra kończy się, gdy któryś z graczy wygra lub gdy plansza zostanie wypełniona (remis).

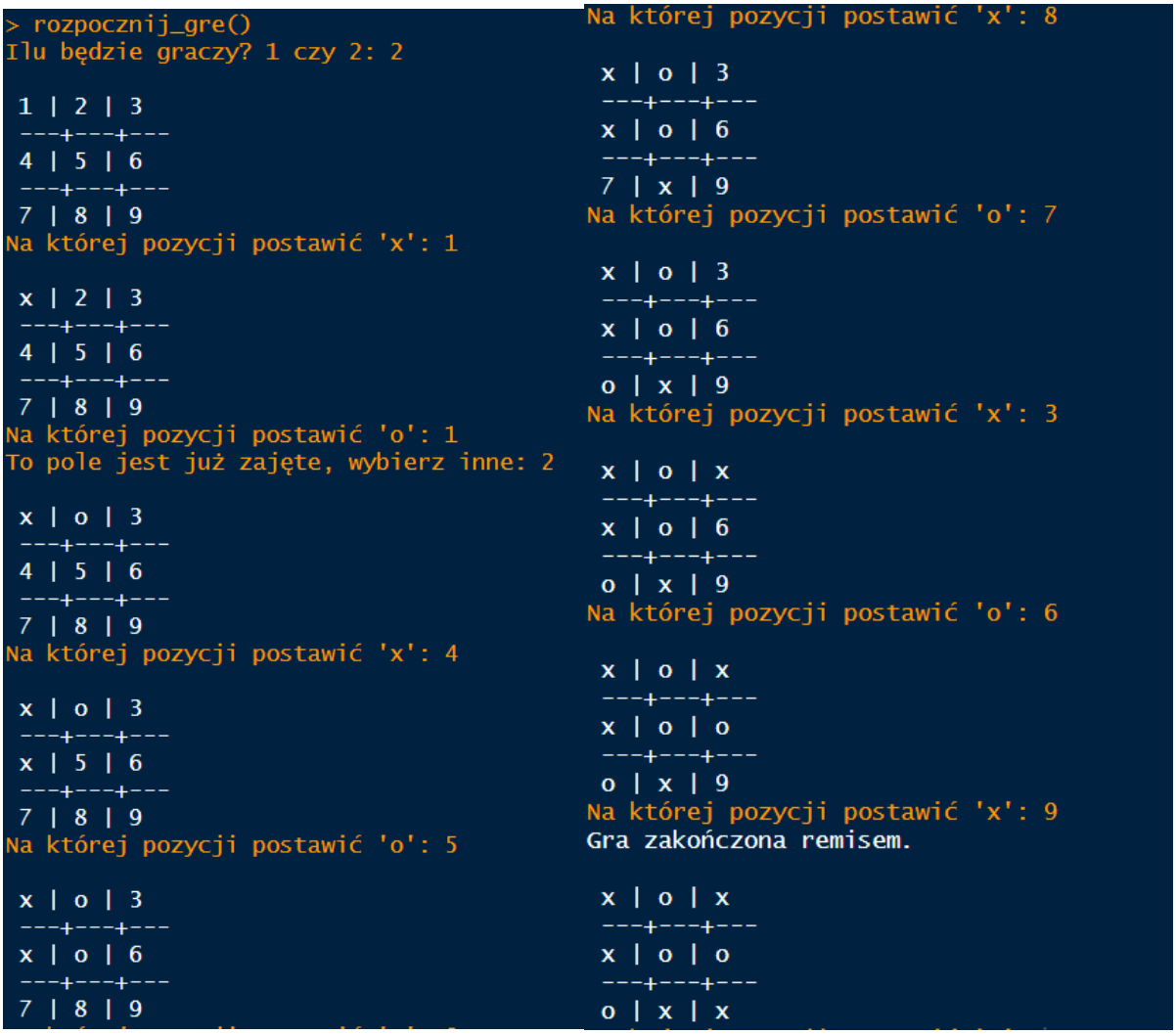
5.4 Testowanie i uruchamianie gry

Gra została przetestowana pod kątem poprawności działania oraz wykrywania sytuacji wygranej, przegranej i remisu. Przeprowadzone testy obejmowały różne scenariusze rozgrywki dla jednego i dwóch graczy.

Pierwsza gra:



Druga gra:


> rozpocznij_gre()
Ilu będzie graczy? 1 czy 2: 2
1 | 2 | 3
4 | 5 | 6
7 | 8 | 9
Na której pozycji postawić 'x': 1
x | 2 | 3
4 | 5 | 6
7 | 8 | 9
Na której pozycji postawić 'o': 1
To pole jest już zajęte, wybierz inne: 2
x | o | 3
4 | 5 | 6
7 | 8 | 9
Na której pozycji postawić 'x': 4
x | o | 3
x | 5 | 6
7 | 8 | 9
Na której pozycji postawić 'o': 5
x | o | 3
x | o | 6
7 | 8 | 9
Na której pozycji postawić 'x': 8
x | o | 3
x | o | 6
7 | x | 9
Na której pozycji postawić 'o': 7
x | o | 3
x | o | 6
o | x | 9
Na której pozycji postawić 'x': 3
x | o | x
x | o | 6
o | x | 9
Na której pozycji postawić 'o': 6
x | o | x
x | o | o
o | x | 9
Na której pozycji postawić 'x': 9
Gra zakończona remisem.
x | o | x
x | o | o
o | x | x

Trzecia gra:

```
> rozpocznij_gre()
Ilu będzie graczy? 1 czy 2: 1
Czy komputer ma grać jako pierwszy, czy jako drugi? 1 czy 2: 1
x gra na pozycji 2

 1 | x | 3
 ---+---+---
 4 | 5 | 6
 ---+---+---
 7 | 8 | 9
Na której pozycji postawić 'o': 1
x gra na pozycji 8

 o | x | 3
 ---+---+---
 4 | 5 | 6
 ---+---+---
 7 | x | 9
Na której pozycji postawić 'o': 4
x gra na pozycji 9

 o | x | 3
 ---+---+---
 o | 5 | 6
 ---+---+---
 7 | x | x
Na której pozycji postawić 'o': 3
x gra na pozycji 7
Komputer wygrywa!

 o | x | o
 ---+---+---
 o | 5 | 6
 ---+---+---
 x | x | x
```

6. Wnioski

6.1 Podsumowanie projektu

Projekt zakończył się sukcesem - stworzono funkcjonalną grę kółko i krzyżyk w języku R, która pozwala na rozgrywkę przeciwko komputerowi lub z drugim graczem. Kod jest czytelny i łatwo rozbudowywalny.

6.2 Możliwość rozwoju projektu

W ramach dalszego rozwoju projektu można rozważyć następujące usprawnienia:

- Implementacja bardziej zaawansowanych algorytmów sztucznej inteligencji, np. algorytmu minimax, aby zwiększyć trudność gry przeciwko komputerowi.

- Dodanie trybu gry wieloosobowej online, pozwalającego na rozgrywkę między graczami zdalnymi.
- Ulepszenie interfejsu użytkownika, np. poprzez dodanie graficznej reprezentacji planszy gry.

7. Wersjonowanie kodu i praca z GIT

7.1 Założenie repozytorium

Ta komenda inicjuje nowe repozytorium Git, tworząc katalog .git.

```
danie@DESKTOP-B7FP92H MINGW64 /e/STUDIA/SEMESTR VI/Usługi sieciowe w biznesie/Pr
ojekt/Gra
$ git init
Initialized empty Git repository in E:/STUDIA/SEMESTR VI/Usługi sieciowe w bizne
sie/Projekt/Gra/.git/
```

7.2 Dodawanie plików do repozytorium za pomocą `git add`

Aby dodać pliki do repozytorium, użyj komendy:

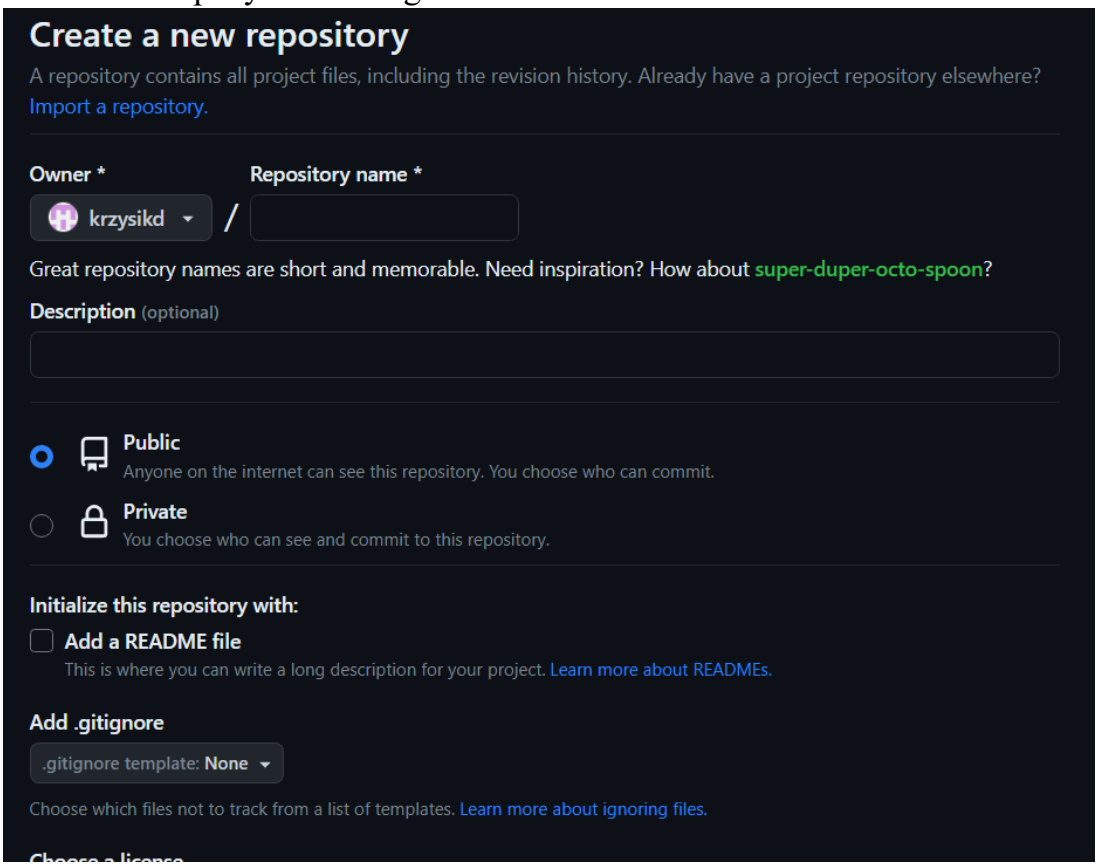
```
danie@DESKTOP-B7FP92H MINGW64 /e/STUDIA/SEMESTR VI/Usługi sieciowe w biznesie/Projekt/Gra (master)
$ git add kolko_krzyzyk.R
```

7.3 Tworzenie commitów za pomocą `git commit`

To krótki opis wprowadzonych zmian.

```
danie@DESKTOP-B7FP92H MINGW64 /e/STUDIA/SEMESTR VI/Usługi sieciowe w biznesie/Projekt/Gra (master)
$ git commit -m "Utworzenie projektu"
[master (root-commit) b555a2f] Utworzenie projektu
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 kolko_krzyzyk.R
```

7.4 Utworzenie repozytorium na github



7.5 Dodanie zdalnego repozytorium

Dodaj zdalne repozytorium, z którym będziesz synchronizować lokalne zmiany:

```
danie@DESKTOP-B7FP92H MINGW64 /e/STUDIA/SEMESTR VI/Usługi sieciowe w biznesie/Pr
ojekt/Gra (master)
$ git remote add origin https://github.com/krzysikd/Uslugi_sieciowe_w_biznesie.g
it
```

7.6 Wysyłanie zmian do zdalnego repozytorium

```
danie@DESKTOP-B7FP92H MINGW64 /e/STUDIA/SEMESTR VI/Usługi sieciowe w biznesie/Projekt/Gra (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 232 bytes | 232.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/krzysikd/Uslugi_sieciowe_w_biznesie.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

7.7 Sprawdzenie historii commitów

Wyświetl historię commitów w repozytorium za pomocą:

```
danie@DESKTOP-B7FP92H MINGW64 /e/STUDIA/SEMESTR VI/Usługi sieciowe w biznesie/Projekt/Gra (master)
$ git log
commit b555a2fe0e6480e8ca2eea0ae92836085073c20a (HEAD -> master, origin/master)
Author: Daniel Krzysik <danielkrzysik00@gmail.com>
Date:   Tue Apr 18 23:19:44 2023 +0200

    Utworzenie projektu
```

7.8 Sprawdzenie statusu repozytorium

Sprawdź status repozytorium, aby zobaczyć, które pliki zostały zmodyfikowane lub oczekują na commit:

```
danie@DESKTOP-B7FP92H MINGW64 /e/STUDIA/SEMESTR VI/Usługi sieciowe w biznesie/Projekt/Gra (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   kolko_krzyzyk.R

no changes added to commit (use "git add" and/or "git commit -a")
```

8. Bibliografia

- https://www.youtube.com/watch?v=Ebe9D5zRkvM&list=WL&index=1&t=2083s&ab_channel=Jaknauczy%C4%87si%C4%99programowania
- https://pbiecek.github.io/Przewodnik/index.html
- Ogólne dostępne materiały w internecie