Київський національний торговельно-економічний університет
Кафедра програмної інженерії та кібербезпеки

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка програмного забезпечення для кіберспортивних змагань»

Студента 2 курсу, 5м групи,
спеціальності
121 «Програмна інженерія»
спеціалізації
«Інженерія програмного забезпечення»

_____
підпис студента

Сперисенка Олександра Валерійовича

Науковий керівник
кандидат технічних наук,
професор

_____
підпис керівника

Пашорін Валерій Іванович

Гарант освітньої програми
доктор технічних наук,
професор

_____
підпис керівника

Криворучко Олена Володимирівна

КИЇВ – 2019

ЗМІСТ

					КНТЕУ 121– 05-19.МР		
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*	*Розробка програмного забезпечення для кіберспортивних змагань*	*Стадія*	*Аркуш*	*Аркушів*
Зав. кафедри		Криворучко О.В.				Зміст	*2*	*48*
Керівник		Пашорін В.І.						
Гарант		Криворучко О.В.			*Зміст*	Факультет обліку, аудиту та інформаційних систем, 2м курс, 5 група		
Розроб.		Сперисенко О.В.						

ВСТУП

Кіберспорт (також відомий як електронний спорт) є формою конкуренції з використанням відеоігор. Найчастіше, кіберспорт приймає форму організованих, багатокористувацьких змагань по відеоіграм, особливо між професійними гравцями, окремо або командно. Хоча організовані онлайн і офлайн змагання вже давно є частиною культури відеоігор, вони були в основному між аматорами аж до кінця 2000-х років, коли участь професійних геймерів і глядачів у цих подіях через потоковий перегляд побачила велику популярність. До 2010-х років кіберспорт був важливим фактором в індустрії відеоігор, і багато розробників ігор активно розробляли ідеї до професійної субкультури електронного спорту.

У 2019 році оцінюється, що загальна аудиторія кіберспортсменів зросте до 454 мільйонів глядачів і що доходи збільшаться до більш ніж 1 млрд. Доларів. Зростаюча доступність потокових мультимедійних онлайн-платформ, зокрема Panda.tv, YouTube і Twitch, стали центральними елементами росту і просування змагань кіберспорт. Повідомляється про аудиторію, яка становить приблизно 85% чоловіків і 15% жінок, причому більшість глядачів у віці від 18 до 34 років. Незважаючи на це, деякі жіночі особистості в кіберспорті сподіваються на збільшення присутності жінок-геймерів. Південна Корея має кілька створених організацій, які мають ліцензію професійних геймерів з 2000 року. Визнання змагань за межами

					КНТЕУ 121– 05-19.МР		
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*	*Розробка програмного забезпечення для кіберспортивних змагань*	*Стадія*	*Аркуш*	*Аркушів*
Зав. кафедри		Криворучко О.В.				Зміст	*3*	*49*
Керівник		Пашорін В.І.						
Гарант		Криворучко О.В.			*Зміст*	Факультет обліку, аудиту та інформаційних систем, 2м курс, 5 група		
Розроб.		Сперисенко О.В.						

Південної Кореї відбулося дещо повільніше. Поряд з Південною Кореєю більшість змагань відбувається в Європі, Північній Америці та Китаї. Незважаючи на великий ринок відеоігор, кіберспорт в Японії відносно слабко розвинені, і це пояснюється значною мірою широкими законами про боротьбу з азартними іграми, які забороняють платні професійні ігрові турніри. Зважаючи на вищевикладені дані, ідеальна аудиторія для кіберспорту є вищі навчальні заклади. Саме в таких навчальних закладах студенти зазвичай цікавляться комп'ютерними іграми, а тому знайомляться з професійним кіберспортом. Наприклад, в Київському національному торговельно-економічному університеті щорічно проводяться кіберспортивні турніри. Для кращого його управління потрібне відповідне програмне забезпечення. Саме тому темою дипломної роботи стало розроблення програмного забезпечення для кіберспортивних змагань.

Актуальність роботи: Часто для проведення кіберспортивного турніру використовуються стороннє програмне забезпечення, керування яким відходить сторонній особі або людині яка малознайома з даним програмним забезпеченням. В такому випадку можливі помилки проведення турніру, через недостатню обізнаність, що може призвести до фінансових втрат. Також не слід виключати втручання сторонньої особи в проведення чесних змагань, що також може призвести до неприємних наслідків. Зважаючи на це слід використовувати перевірене, просте, зручне та надійне програмне забезпечення, яке збереже вас від неприємних наслідків.

Об'єкт дослідження: кіберспорт та кіберспортивні змагання.

Предмет дослідження: створення алгоритмів та їх реалізація на операційній системі Windows з використанням мови програмування C# у вигляді програмного продукту "Жеребкування" за допомогою технологій об'єктно-орієнтованого програмування.

Мета роботи: розробка та реалізація алгоритмів та інтерфейсів для створення програмного продукту "Жеребкування" для операційної системи Windows.

Методи і технології дослідження: аналіз алгоритмів; візуальне і об'єктно-орієнтоване програмування; створення додатку для операційної системи Windows з використанням графічної бібліотеки WinForms.

Завдання дослідження:

- аналітичний огляд предметної області;
- написання технічного завдання;
- розгляд графічного інтерфейсу WinForms;
- розгляд створення додатку для операційної системи Windows на мові програмування C#;
- розробка алгоритмів для додатку «Жеребкування»;
- розробка додатку «Жеребкування» для ОС Windows;
- тестування створеного додатку.

Наукова новизна дослідження: розробка програмного забезпечення для операційної системи Windows з використанням графічної бібліотеки WinForms для використання на кіберспортивних змаганнях, що має покращенний метод жеребкування олімпійської системи.

Результати роботи: Створення додатку "Жеребкування" для операційної системи Windows з використанням графічного інтерфейсу WinForms на мові програмування C#.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Розвиток та становлення кіберспортивних змагань

Комп'ютерні ігри, як відомо, стали частиною сучасного способу життя, і в цивілізованому світі дуже мало людей, які могли б уявити повсякденне життя без них. Молодше покоління проводить багато часу за допомогою комп'ютерів; старше покоління в найкращому випадку не втручається. У гіршому - бурчить і намагається боротися з цим як із річчю. Але так чи інакше є занадто мало людей, які вважають комп'ютерні ігри чимось вартим їх уваги (за винятком розробників та великих дистриб'юторів ігор).

Однак часи змінюються. Кіберспорт увійшов у життя геймерів. Те, що розпочалося дружньою вечіркою десь у підвалі, переросло у цілу галузь, промисловість, яка забезпечує емоції та розваги. Промисловість, яка росте з кожним роком, приносить не тільки нових гравців, а й великі грошові кулі.

Прийнято говорити, що вихідним пунктом для кіберспорту став турнір Red Annihilation Quake, який відбувся в 1997 році на виставці Electronic Entertainment Expo під керівництвом Джона Кармака. Вісім учасників змагалися за приз у 5 тисяч доларів та Ferrari 328 GTS. Перемогу здобув Денніс «Треш» Фонг, який вважається першим професіоналом у кіберспорті.

Говорячи про цю справді історичну перемогу, Тім Вілліц, директор студії та колишній співвласник розробника відеоігор Id Software (Doom, Quake), сказав: «Він створив фундамент, який допоміг нам розвинути кіберспорт як галузь в близькому майбутньому».

Після цього ситуація лише розвивалася та прогресувала. Конвенції щодо ігор стали більшими, і так же ріс інтерес. Невдовзі, у 2000 році, в Йонгіні, Південна Корея, відбулися перші Всесвітні кіберспортивні ігри(World Cyber Games). Їх спонсорували Samsung та Microsoft, а також

підтримували Міністерство культури і туризму та Міністерство інформації та комунікацій Південної Кореї.

Ці заходи, які були організовані під прапорами WCG, намагалися слідувати Олімпійським іграм. Це включало офіційну церемонію відкриття, гравці різних країн змагалися за грошові премії, золоті, срібні та бронзові медалі. У WCG також був власний девіз «Поза грою(Beyond the game)». У ньому також були свої дисципліни: Quake III Arena, FIFA 2000, Age of Empires II, StarCraft: Brood War та Unreal Tournament. Загальний призовий фонд склав 200 тис. Доларів, а в змаганнях взяли участь 174 конкуренти з 17 різних країн (*рис. 1.1*).

Крім того, WCG проводився щорічно, і призові фонди, дисципліни та кількість учасників змагань лише зростали. Пізніше були додані такі ігри, як Counter Strike, Warcraft, Need for Speed... Що стосується самого WCG, то вони відбувалися не лише в Південній Кореї, а й у США, Італії, Німеччині, Китаї.

Рис.1.1. WCG 2004

Але є не тільки WCG у світі кіберспорту. Є також американська вища ліга ігор (Major League Gaming (MLG)), яка спеціалізується на турнірах Call

of Duty і славиться першою телевізійною трансляцією турніру з відеоіграми (це була трансляція популярної гри в США - Halo) та Intel Extreme Masters (IEM), де Counter Strike та Warcraft були основними дисциплінами протягом тривалого часу.

Саме тоді, в середині 2000-х років, було сформовано більшість команд, які представляли геймерів у професійному плані. Серед них досить відомі та багаті організації, такі як OpTic Gaming, Evil Geniuses, Team SoloMid, eRa Eternity, Cloud9, Fnatic, Mineski, Counter Logic Gaming, SK Telecom T1, Splyce, Team EnVyUs, Natus Vincere та інші.

Тож ви можете подумати - це вихідна точка кіберспорту, адже це був момент часу, коли бізнес вступив в кіберспорт. Сформувалась точка зору, що можна було б вигідно збирати людей, змушувати їх грати разом, тренуватися разом, змагатися та перемагати. Візьміть перемоги не тільки для самооцінки, але і для того, щоб виграти гроші, приносити дохід. Це був час перших інвесторів, перших успіхів та перших невдач. Але справжня революція ще попереду. Успіх був у повітрі, але все-таки занадто далеко, щоб називати це переломною точкою.

Справжня революція, яка все перевернула, розпочалася в 2011 році. Все сталося завдяки двом головним подіям у світі ігор. Без них ви не уявляєте сучасних електронних видів спорту.

Перша і основна подія - це запуск потокової платформи, яка спеціалізувалася на трансляціях відеоігор - Twitch.tv. Якщо говорити про цей випадок, то це був насправді не старт нової платформи, це було більше схоже на відсторонення від більшого сервісу Justin.tv. Причини були прозаїчними - ігрова секція мала такий темп зростання, що не було сенсу підтримувати інших поряд. Тож генеральні директори вирішили відокремити ігровий контент. Тільки за п’ять днів з моменту запуску Twitch отримав 6,5 мільйонів переглядів при загальній аудиторії 60-70 тисяч. З тих пір вона постійно зростала. У 2012 році Twitch був підтриманий значними інвестиціями венчурного капіталу, у 2013 році - 15 мільйонів доларів та 20 мільйонів доларів. Громада та аудиторія зростали, а також доходи. Twitch здобув лідируюче місце серед потокових платформ і навіть отримав достатньо велику роль, щоб стати конкурентом Youtube.

Після свого стрімкого підйому Twitch став сервісом №1 в грі-стрімінгу. League of Legends, Counter-strike Global Offtack, Dota 2 та Call of Duty швидко

стали популярними іграми, замінивши класику на зразок Quake, Starcraft у середньому житті гравця (*рис. 1.2*).

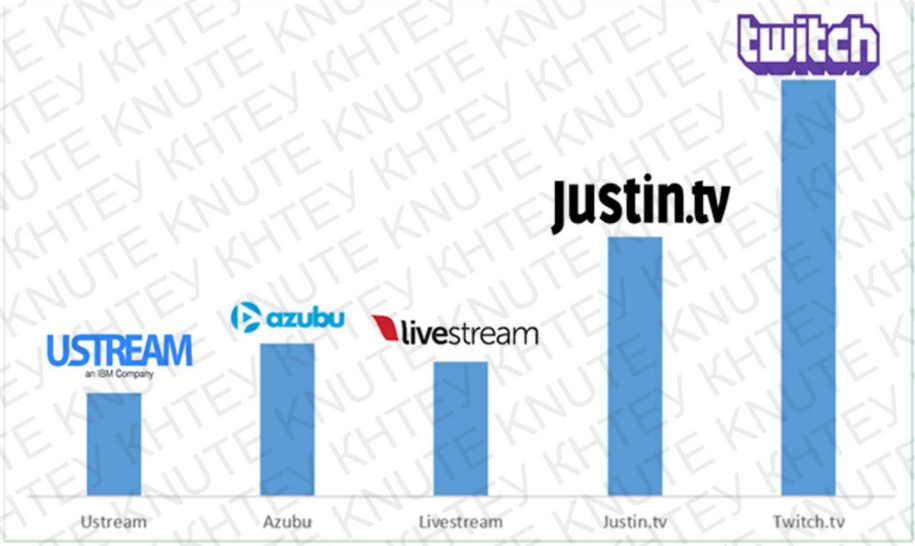

Рис.1.2. Пропорційна аудиторія потокових сервісів(2011)

Аудиторія продовжувала експоненційний ріст. Все через те, що Twitch став місцем, куди звичайні геймери заходили, щоб робити трансляцію свого ігрового процесу, а також геймерів, які заходили сюди, щоб подивитися їх. Побачити, як інші грають у свої ігри, як вони взаємодіють з нею, як вони виграють чи провалюються в різних ситуаціях. Побачити реакції, коли вони «роблять» гру, або зазнають поразки потужним ворогом. Деякі займалися подібними трансляціями серйозно, деякі робили їх «просто для забави» (just for lulz) та взаємодії з глядачами в чаті. Головне, що зробило Twitch популярним - це «підписка», яка дозволила глядачам взаємодіяти із стримерами безпосередньо, можливість стримерів отримувати деякі «преміальні» функції для канала, а також дала можливість монетизувати свої трансляції не лише за допомогою реклами, але також безпосередньо від користувачів. Завдяки Twitch ігри стали джерелом доходу для таких людей. Стабільний дохід, який був досить приголомшливим і революційним на той час.

Для демонстрації, топ-стрімер League of Legends Майкл "Imaqtpie" Сантана заробив 2 мільйони доларів за рік у 2015 році, лише від потокового

потоку, не враховуючи доходу від спонсорства чи маркетингу. Це в середньому 30 тисяч доларів на місяць лише за гру.

Але потоки відеоігор були не єдиною причиною успіху. Вся справа в тому, що Twitch.tv також розпочав трансляцію кращих ігрових турнірів. Чемпіонати Ліги Легенд (League of Legends) зробили найбільший успіх. Наприклад, на чемпіонаті, який проходив у 2011 році, було 1,7 мільйона глядачів, і цей показник збільшився до 8,2 мільйона у 2012 році. Для чемпіонату 2013 року загальна кількість глядачів зросла до 32 мільйонів. І це лише чемпіонат цієї гри. Наприклад, церемонія інавгурації Трампа мала "лише" 30,6 мільйона глядачів.

То яка причина, що стільки геймерів прив'язуються до своїх комп'ютерів, дивляться трансляції, підтримують команди і навіть платять за це реальні гроші?

Для відповіді ми переходимо до другої події, яка дала гарний поштовх для електронних видів спорту як галузі та зробила кіберпрофесію річчю. Що це було? Популярні ігри на Twitch були не тільки дружніми до користувача та зручними для доступа, вони були багатокористувацькі, такі в які можна грати з друзями через Інтернет. Ігри, що використовують безкоштовні або free-to-play бізнес-моделі з їх ігровим процесом, зосередженим на швидких змагальних командних матчах, принесли також багато гравців, враховуючи низький рівень входу. У цих величезних громадах аматори переросли в професіоналів, граючи на найвищих рівнях.

Перший Dota International, який влаштував Valve в Кельні в 2011 році, став відправною точкою, після якої електронні види спорту та кіберспорту змінилися на сучасну форму. Справжня спортивна дисципліна. Чому так? Тому що International змінив організацію чемпіонатів такою, якою є. Він встановив тенденції та стандарти таких подій у майбутньому.

Valve надав величезний приз призові - 1,6 мільйона доларів з призом в мільйон доларів переможцю конкурсу. 16 команд DotA з усього світу приїхали в Кельн за шанс виграти головний приз. Решта 600 тисяч були розподілені між конкуруючими командами відповідно до зароблених ними місць. Зауважимо, першим в історії переможцем DotA Internationale стала українська команда Natus Vincere, яка забрала важко зароблений мільйон додому (*рис. 1.3*).

Рис. 1.3. Українська кіберспортивна команда з чеком на 1 млн. дол.

Пізніше Valve зняв документальний фільм про ті події під назвою «Free to Play», де вони критично подивилися на життя кількох гравців та команд, які брали участь у турнірі. Це було головним чином зосереджено на житті членів команд і на тому, як їх прихильність до DotA вплинула на них. Після першого міжнародного турніру кількість глядачів зросла, а також зріс призовий фонд (*рис. 1.4*).

Riot Games та їх League of Legends не відстають у своїх зусиллях у кіберспорті. І якщо чемпіонат першого сезону був скромним і не дуже помітним (призовий фонд склав «всього» 100 тис. доларів США, 8 команд-конкурентів, що представляли 4 регіони), другий сезон чемпіонату, який відбувся у 2012 році, був досить глобальним. Цього разу призовий фонд збільшився до 2 мільйонів доларів (1 мільйон для переможця, такий же, як і Valve), і турнір проходив у Лос-Анджелесі. Після проходження кваліфікації у відповідних регіонах 12 команд із 6 різних регіонів (Північна Америка, Європа, Китай, Тайвань, Південна Азія та Корея) змагалися за гроші.

Рис. 1.4. Призовий фонд The International

Чемпіонат світу в другому сезоні мав 8,2 мільйона глядачів (максимум 1,1 мільйона глядачів одночасно), що робить Чемпіонат світу 2 сезону найпопулярнішою подією кіберспорту у той час. Наприклад, королівське весілля у Британії того року мало лише 300 тис. глядачів, а YouTube на літніх Олімпійських іграх 500 тис. глядачів.

League of Legends побила власний рекорд наступного року, коли квитки на фінал, який відбувся в Стейплс-Центрі, були розпродані лише за годину після початку продажів.

Ці новини потрапляють на головні сторінки не лише на порталах, пов’язаних із іграми, але й на більш відомих медіа у "широкому світі". Люди нарешті почали розуміти, що електронні види спорту є значно більшими та впливовішими, ніж вони коли-небудь уявляли.

Ігри стали не лише можливістю приємно провести час, а й чимось більшим. Вони перетворилися на кар’єрну можливість.

І це не просто обговорюється - це робиться. США визнали гравців електронних видів спорту спортсменами. Більше того, підлітки вибирають між коледжем та кібер-службою, і часто вони обирають електронні види спорту. Тому що це реальна можливість, щоб їхні навички були нагороджені славою та грошима. Це не просто субкультура для підлітків, які збираються разом, вона стала розважальним бізнесом, таким же як футбол, хокей чи гольф, де все більше спортсменів поділяють таку думку. Навіть Fortune і Forbes почали писати про те, як вигідно інвестувати в електронні види спорту.

Є компанії, які проводять щорічні чемпіонати та турніри, які вражають своєю складністю, масштабом та призовими фондами. Є також команди, які мають професійних геймерів за контрактами, із зарплатою, проживанням та харчуванням своїх молодих співробітників. Все з однієї причини - щоб грати і перемагати, заробляти гроші та славу. Для отримання доходу. Зараз E-sports спонсорується не лише його "традиційними" партнерами, такими як ігрові установки чи компанії з комп'ютерних технологій, але й іншими "великими світовими" інвесторами.

Наприклад, League of Legends, починаючи з сезону 2018 року, буде «франчайзинговою» лігою. Команди, які виступають у північноамериканській лізі Riot, будуть бізнес-проектами з повними інвесторами. Більшість команд матимуть інвесторів команди НБА за спиною, з величезними бюджетами та великими інвестиціями. У європейській лізі в електронні види спорту вклалися також великі організації, такі як PSG, Schalke 04, FC Copenghagen, Red Bull, Valencia CF. І все це лише початок.

Компанія з маркетингової розвідки під назвою Newzoo, яка спеціалізується на аналізі електронних видів спорту, дослідила прогнози доходів, і наведені цифри нічим не вражають. Згідно з повідомленнями, до 2020 року дохід та кількість глядачів збільшаться вдвічі (*рис. 1.5*).

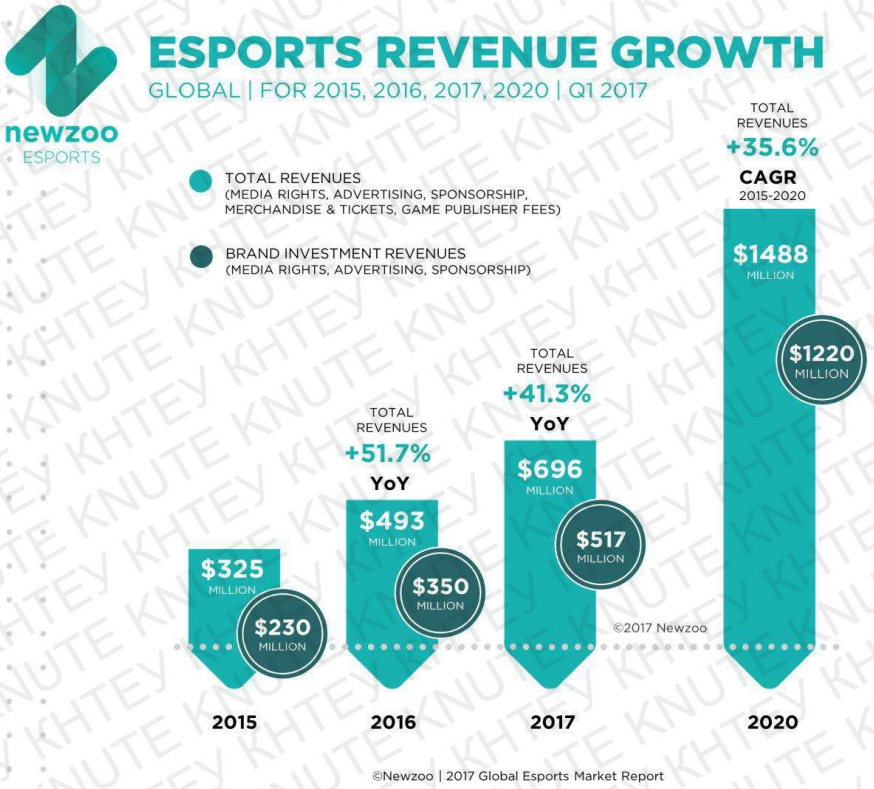

Рис. 1.5. Дохід кіберспорту (по версії Newzoo)

Просто спробуйте уявити ці цифри. Весь дохід, кількість глядачів, зростання інтересів та симпатій досягнуть висот, завдяки яким NBA та NHL заздрять тому, що відбувається в електронних видах спорту. Тож не дивно, що все більше «акул» великих грошей розглядають електронні види спорту як смачну частину торта. Перспективи більш ніж очевидні.

1.2. Опис предметної області

З недавніх пір словосполучення комп'ютерні ігри міцно увійшло в наше життя, кожен, хто має комп'ютер напевно зміг відчути їх привабливість, мабуть, гра закладена в саму природу людини з найдавніших часів - вистежити звіра, заманити його в пастку - це теж свого роду гра. Але тепер ми позбавлені цього в житті, а інстинкти залишилися і вони знаходять свій вихід в комп'ютерних іграх. Саме томе з’являються змагання в іграх, адже, як і раніше кожен хоче бути кращим. Так було засновано кіберспортивні

змагання, що на даний час є дуже популярним та шалено привабливим для інвесторів заняттям.

Найбільш поширеними жанрами відеоігор, пов'язаними з кіберспортом, є багатокористувацька онлайн-бойова арена (MOBA), шутер від першої особи (FPS), бойові дії, цифрові колекційні карткові ігри і стратегії реального часу (RTS). Популярні назви кіберспорт включають ігри MOBA, такі як, League of Legends, Dota 2 і Smite; FPS, такі як Counter-Strike і Call of Duty, CrossFire і Rainbow Six Siege, які знаходяться в суб-жанрі FPS тактичних шутерів, Overwatch, в суб-жанрі FPS геройського-шутера; бойових ігор, таких як Street Fighter, Super Smash Bros., Mortal Kombat; цифрові колекційні карткові ігри, такі як Hearthstone; а також RTS StarCraft. Нещодавно з'явилися змагання, структуровані подібно до американських професійних видів спорту, з платними гравцями та серіями регулярних і плей-офф. Легітимність кіберспорту як спортивного змагання залишається під питанням; однак, кіберспорт стоїть на рівні з традиційними видами спорту на багатонаціональних заходах, а Міжнародний олімпійський комітет вивчає питання включення їх до майбутніх олімпійських заходів.

Часто для проведення кіберспортивного турніру використовуються стороннє програмне забезпечення, керування яким відходить сторонній особі або людині яка малознайома з даним програмним забезпеченням. В такому випадку можливі помилки проведення турніру, через недостатню обізнаність, що може призвести до фінансових втрат. Також не слід виключати втручання сторонньої особи в проведення чесних змагань, що також може призвести до неприємних наслідків. Зважаючи на це слід використовувати перевірене, просте, зручне та надійне програмне забезпечення, яке збереже вас від неприємних наслідків.

Існує декілька варіантів жеребкування, а саме: олімпійська система, кругова система, система з вибуванням після двох поразок та швейцарська система.

Олімпійська система (Single Elimination) в спортивних змаганнях - система розіграшу, при якій учасник вибуває з турніру після першого ж програшу (за підсумками однієї гри або серії з декількох ігор між двома учасниками, що дозволяє однозначно визначити безумовного переможця). Забезпечує виявлення переможця за мінімальне число турів і сприяє напруженій боротьбі в турнірі.

Порядок розіграшу. Число учасників турніру має бути ступенем двійки (8, 16, 32, 64, 128 і так далі). Якщо число учасників відрізняється від правильного, можна застосовувати цю схему турніру з додаванням фіктивних («порожніх») учасників, до найближчого правильного кількості. Учасник, якому випадає гра з «порожнім» (зазвичай спортсмен, з більш високим рейтингом), отримує в цьому турі технічну перемогу. Практично застосування даної схеми зручно при числі учасників, рівному в точності 8, 16, або перевищує 20. При менш ніж 8 учасників краще грати турнір за коловою системою (число партій буде прийнятним).

Двійковий логарифм числа учасників визначає число кіл розіграшу (турів): для 2 учасників - один, для 4 - два, для восьми - три, для 16 - чотири. Загальна кількість ігор на одиницю менше числа учасників. Кола розіграшу зазвичай називаються за кількістю пар учасників: для 1 пари - «фінал» (він визначає переможця), для 2 пар - «півфінал», для 4 пар - «чвертьфінал», для 8 пар - «одна восьма фіналу», для 16 пар - «одна шістнадцята фіналу» і так далі. У кожному колі з учасників складаються пари, які відіграють між собою (це може бути одна гра, або матч з декількох ігор, в якому перемагає набрав більше очок; принципово важливо, що результат туру завжди певний - нічиїх

бути не може). З кожної пари в наступне коло виходить переможець, а переможений вибуває з турніру. Учасник, який виграв фінальний круг, стає переможцем, його останній суперник отримує друге місце. Якщо регламент турніру вимагає присвоєння і третього місця, то проводиться додатковий матч за нього між двома учасниками, які програли в півфіналі.

Кругова система (Round Robbin) - в спортивних змаганнях система розіграшу, при якій кожен учасник турніру грає з кожним в ході туру або раунду. Популярна в багатьох ігрових видах спорту, особливо в національних чемпіонатах і при відбіркових турнірах до чемпіонатів світу або континентів. Вважається найбільш справедливою, але при цьому вимагає найбільшого числа ігор для розподілу місць, в порівнянні з іншими турнірними системами.

Порядок розіграшу. Порядок зустрічей супротивників один з одним при круговій системі не має великого значення. Але учасники в парі чергового туру зазвичай визначаються жеребкуванням. Іноді ж порядок зустрічей призначають так: привласнюють кожному з N учасників змагань порядкові номери, від 1 до N. Якщо число учасників парно, тобто N = 2K, то записують в правій колонці зверху вниз номера від 1 до K зверху вниз, а в лівому стовпці номера від K + 1 до N від низу до верху. Учасники, номери яких написані навпроти один одного, зустрічаються в першому турі. Для складання аналогічних таблиць для наступних турів номер 1 залишають на місці, а всі інші номери пересувають проти годинникової стрілки, кожен раз на один крок. У зв'язку з останньою процедурою кругова система проведення змагань, власне, і називається круговою. Якщо число учасників непарній, то додають номер нуль, і для отриманого набору номерів, кількість яких тепер парно, виконують вищезазначені процедури. Учасник, проти номера якого в будь-якому турі варто число 0, в цьому турі вільний, не грає.

Кількість зустрічей при круговій системі визначається за формулою 0,5 × N × (N-1), де N кількість гравців. Кількість турів (за наявності технічної можливості одночасного проведення достатнього числа ігор) одно для парного числа учасників і для непарного (в останньому випадку кожен учасник пропускає один тур, в якому йому не знаходиться суперника).

За результатами кожної гри учаснику нараховується певна кількість очок. Наприклад, в шахах традиційно нараховують 1 очко за виграш, 0 очок за програш і 0,5 очка за нічию. Очки, набрані учасниками протягом усього турніру, підсумовуються. Місця розподіляються по спадаючій кількості набраних очок.

Якщо двоє або більше учасників набрали однакову кількість очок, з метою рівномірного зменшення місць застосовуються додаткові критерії: коефіцієнт Бергера, результат особистої зустрічі, уточнений результат ігор (наприклад, в хокеї і футболі може застосовуватися різниця числа забитих і пропущених голів - у кого вона краще, той отримує більш високе місце). Правила турніру визначають застосовувані в ньому додаткові критерії розподілу місць у учасників з рівним числом набраних очок. Якщо за всіма критеріями учасники виявляються рівними, правила можуть передбачати або проведення між ними додаткових зустрічей до певного позитивного результату, або «поділ місць», коли рівні учасники вважаються одночасно зайняли два або більше місць в підсумковій таблиці.

Система з вибуванням після двох поразок (Double Elimination)

Олімпійська система з вибуванням після двох поразок, жарг. «Мінуска» (англ. Double Elimination) - система проведення турнірів, в якій учасник вибуває з турніру після двох поразок. В цьому і є відмінність від простої олімпійської системи, в якій тільки одна поразка призводить до вильоту.

Початкові умови. Число учасників турніру має бути ступенем двійки (8, 16, 32, 64, 128 і так далі). Якщо число учасників відрізняється від правильного, можна застосовувати цю схему турніру з додаванням фіктивних («порожніх») учасників, до найближчого правильного кількості. Учасник, якому випадає гра з «порожнім», отримує в цьому турі технічну перемогу. Практично застосування даної схеми зручно при числі учасників, рівному в точності 8, 16, або перевищує 20. При менш ніж 8 учасників краще грати турнір за коловою системою (число партій буде прийнятним).

Турнір розділений на дві сітки - верхню і нижню (сітку переможців і сітку тих, хто програв). Всі учасники починають турнір у верхній сітці. Учасники розбиваються на пари, які проводять в першому турі гри між собою.

Формування пар на початку турніру може відбуватися:

- жеребкуванням;

- за принципом сильний проти слабкого на підставі рейтингів або результатів попередніх змагань (в разі наявності "порожніх" осередків в сітці змагань найсильніший гравець ставиться в пару з "порожнім" учасником);

- посівом, здійснюваним суддівською колегією або організаторами в присутності незалежних представників (посів може здійснюватися виходячи з рейтингів учасників, місця їх проживання, віку та ін.).

Ігри у верхній і нижній сітках. На кожен тур у верхній сітці в нижній проводиться два: Перший (з пари) тур нижньої сітки проводиться одночасно з туром верхньої: грають між собою гравці, що залишилися в сітці після попереднього туру. Ті, хто програв в нижній сітці вибувають з турніру. Ті, хто програв у верхній сітці переходять в нижню. В результаті у верхній сітці стає вдвічі менше гравців, а в нижній їх число залишається колишнім (половина вибула, половина перейшла з верхньої сітки).

Складаються пари з тих, хто переміг в попередньому турі нижньої сітки і тих, хто перейшов з верхньої. Між ними проводиться ще один тур на вибування. Верхня сітка в цей час не грає. В результаті в нижній сітці залишається стільки ж гравців, скільки у верхній.

По завершенні кожного туру верхньої сітки і відповідної пари турів нижньої в обох сітках виявляється рівне число гравців. Після фінальних турів в обох сітках виявляється по одному гравцеві, вони і грають між собою суперфінал. Перевага даної системи в тому, що вона працює однаково при будь-якій кількості учасників, недолік - в більшій кількості партій, а також велика різниця в числі партій, які повинен зіграти учасник для досягнення фіналу по нижній і верхній сітці.

Як правило, в чистому вигляді така сітка практично не використовується. З заздалегідь визначеного туру гравці верхньої і нижньої сіток об'єднуються, і фінальна частина турніру проходить по системі до однієї поразки. При цьому переможці зустрічей верхньої сітки зустрічаються з переможцями зустрічей нижньої. Для визначення пар, між переможцями нижньої сітки проводиться додаткова жеребкування.

Суперфінал. Між двома фіналістами турніру проводиться фінальний тур, який може також називатися «суперфіналом» (тоді останні матчі в кожної з сіток називаються «фіналами»). Фінальний тур проводиться в одну або дві гри, його мета - домогтися, щоб у одного з фіналістів число поразок в турнірі досягло двох. Якщо переможець верхньої сітки виграє першу гру, то він отримує перше місце, а його суперник - друге. Якщо переможець верхньої сітки програє першу гру, проводиться друга гра, переможець якої отримує перше місце, той, хто програв - друге. Такий варіант називають «Full Double Elimination».

Швейцарська система - система проведення спортивних турнірів. Особливо поширена в інтелектуальних іграх, таких як шахи, шашки, сьогі, го, рендзю і їм подібних. Вперше була застосована на шаховому турнірі в Цюріху (Швейцарія) в 1895 році, звідки і отримала свою назву. Турнір проходить без вибування, в кожному турі, починаючи з другого, пари суперників відбираються так, щоб зустрічалися між собою учасники, що набрали рівну кількість очок. За цей рахунок з турніру виключаються партії між свідомо непорівнянними за силою противниками, що дозволяє для визначення переможців обійтися невеликим, порівняно з круговою системою, числом турів при великому числі учасників.

Умови застосування. Традиційно для отримання найбільш об'єктивного результату турніри проводилися за коловою системою, в якій кожен учасник грає не менше однієї гри з кожним і переможець визначається за сумою набраних очок. Але в коловою системою зі збільшенням числа учасників необхідне число зустрічей швидко зростає, тому її застосування при кількості учасників більше двох-трьох десятків стає нереальним. У турнірах, що проводяться за швейцарською системою, іноді беруть участь понад сто гравців - якщо в коловою системою 100 гравцям треба було б 4950 зустрічей в 99 турах, то в швейцарській досить 450 партій в 9 турах (виграш в одинадцять разів).

Швейцарська система дозволяє зменшити витрати часу за рахунок того, що по ній грають деякий заздалегідь визначене положенням про турнір кількість турів, а система підбору пар для кожного туру організована так, щоб забезпечити в результаті впевнене розподіл місць згідно набраними очками. Вважається, що для виявлення переможця досить стільки турів, скільки необхідно ступенів для виявлення переможця в нокаут-системі при тій же кількості учасників. За деякими оцінками [1], при N учасників {\

displaystyle {\ sqrt {N + 2k}}} {\ sqrt {N + 2k}} турів справедливо розставляють k + 1 перших гравців, на практиці застосовують формулу {\ displaystyle \ log _ {2} N + \ log _ {2} k} {\ displaystyle \ log _ {2} N + \ log _ {2} k}, округляючи при обчисленнях значення обох логарифмів до найближчого цілого. Загальна кількість зустрічей визначається формулою M * N / 2, де N - кількість гравців (парне) і M - кількість турів (коли всі гравці грають у всіх турах).

Порядок проведення турніру. У першому турі всі гравці упорядковуються (випадковим жеребкуванням, або по рейтингу). Пари складаються за принципом: перший з верхньої половини таблиці з першим з нижньої половини, другий - з другим, і так далі. Якщо, наприклад, в турнірі 40 учасників, то перший грає з 21-м, другий з 22-м і т. Д. При непарному числі гравців гравець, який має останній номер, отримує очко без гри.

У наступних турах всі гравці розбиваються на групи з однаковою кількістю набраних очок. Так, після першого туру груп буде три: виграли, програвши і зіграли внічию. Якщо в групі виявляється непарну кількість гравців, то один гравець перекладається в наступну очкову групу.

Пари гравців для наступного туру складаються з однієї очкової групи з такого самого, що і в першому турі, рейтинговим принципом (кращий гравець з верхньої половини групи по можливості зустрічається з кращим гравцем з нижньої половини цієї групи). При цьому, однак, не допускається, щоб одна і та ж пара грала в турнірі більше однієї гри. При грі в шахи або шашки, крім того, діє правило чергування кольору: бажано, щоб у кожного учасника від туру до туру чергувався колір фігур (щоб гравець мав рівну кількість ігор білими і чорними), в будь-якому випадку не допускається три партії поспіль (в шашках - чотири) одним кольором, крім останнього туру.

При непарному числі гравців гравець, який має останній номер в останній очкової групі (з ще не отримували очко за пропуск), отримує очко без гри.

Місця в турнірі розподіляються за набраним кількістю очок. Учасники, що набрали рівну кількість очок, зазвичай розподіляються за коефіцієнтом Бухгольца, який визначається як сума очок, набраних усіма суперниками даного гравця в турнірі або за коефіцієнтом Солкофа, який визначається як сума очок, набраних усіма суперниками даного гравця в турнірі, виключаючи найкращий і самий найгірший результати. Крім них (або разом з ними) може застосовуватися середній рейтинг суперників (тому, у кого суперники мають більш високий середній рейтинг, присуджується більш високе підсумкове місце) або так званий «коефіцієнт прогресу» - більш високе місце отримує гравець, який по ходу турніру довше знаходився на більш високому місці, ніж набрав однакову кількість очок суперник).

1.3. Висновки до розділу 1

Кіберпростір стрімко розвивається, так же прогресує й розважальна складова інтеренету. Це породжує багато розважального контенту, а саме ігор. Гравці змагаються між собою на кіберспоривних змаганням, які вже не є лише розвагою, це приносить величезні прибутки(згадаємо перших в історії переможців DotA Internationale, українську команду Natus Vincere, яка виграла мілліон долларів, а також мільярдні прибутки компаній-партнерів DotA Internationale).

У 2019 році оцінюється, що загальна аудиторія кіберспортсменів сягає близько 500 мільйонів глядачів. Повідомляється, що аудиторія складається приблизно з 85% чоловіків та 15% жінок, причому більшість глядачів у віці від 18 до 34 років. Зважаючи на це, ідеальна аудиторія для кіберспорту є вищі навчальні заклади. Саме в таких навчальних закладах студенти зазвичай

цікавляться комп’ютерними іграми, а тому знайомляться з професійним кіберспортом. Наприклад, в Київському національному торговельно-економічному університеті щорічно проводяться кіберспортивні турніри. Для кращого його управління потрібне відповідне програмне забезпечення.

Зазвичай для проведення кіберспортивного турніру використовуються стороннє програмне забезпечення, керування яким відходить сторонній особі або людині яка малознайома з даним програмним забезпеченням. В такому випадку можливі помилки проведення турніру, через недостатню обізнаність, що може призвести до фінансових втрат. Також не слід виключати втручання сторонньої особи в проведення чесних змагань, що також може призвести до неприємних наслідків. Зважаючи на це слід використовувати провірене, просте, зручне та надійне програмне забезпечення, яке збереже вас від неприємних наслідків. Саме тому темою дипломної роботи стало розроблення програмного забезпечення для кіберспортивних змагань.

РОЗДІЛ 2
ВИКОРИСТАНІ ІНСТРУМЕНТИ ТА ПРОЕКТУВАННЯ

2.1. Постановка завдання

Завданням даної роботи є розробка додатку для жеребкування команд, що буде використовуватися на кіберспортивних змаганнях.

В програмному забезпеченні важливо не тільки якість, а й безпека, тому краще використовувати свій програмний продукт, або той в якому ви впевнені. Програмний продукт даної роботи пропонує спеціально розроблену програму для КНТЕУ. Це передбачає безпеку, чесність проведення змагання, а також зручність та адаптивність під вимоги навчального закладу. Програмне забезпечення написано на мові програмування C# та виконується на будь-якому комп'ютері з операційною системою Windows. Програмний продукт має функціонал для жеребкування, тобто розподілення команд по парам для подальшого визначення переможця. Програма здатна будувати турнірну таблицю, де буде відображено переможець, з кожним етапом жеребкування таблиця буде адаптивно змінюватися. Противники визначаються в випадковому порядку, їх кількість вводиться на початку. Має використовуватися Олімпійська система (Single Elimination).

2.2. Графічна бібліотека Windows Forms

Windows Forms (WinForms) - це графічна бібліотека (GUI), що входить до складу Microsoft .NET Framework або Mono Framework, що забезпечує платформу для запису багатих клієнтських додатків для ноутбуків та

настільних і планшетних ПК. Незважаючи на те, що WinForms розглядається як заміна для більш ранньої та більш складної бібліотеки класів Microsoft Foundation, заснованої на C ++, вона не пропонує порівнянну парадигму і діє лише як платформа для рівня користувальницького інтерфейсу у багаторівневому рішенні.

На заході Microsoft Connect 4 грудня 2018 року Microsoft оголосила про випуск Windows Forms як проект з відкритим кодом на GitHub. Він випускається під ліцензією MIT. З цим випуском Windows Forms стала доступною для проектів, орієнтованих на фреймворк .NET Core. Однак фреймворк все ще доступний лише на платформі Windows, а неповна реалізація WinForms Mono залишається єдиною кросплатформою.

Додаток Windows Forms - це програма, керована подіями, що підтримується .NET Framework від Microsoft. На відміну від пакетної програми, вона витрачає більшу частину свого часу, просто чекаючи, коли користувач щось зробить, наприклад заповнити текстове поле або натиснути кнопку.

Форми Windows надають доступ до загальних елементів внутрішнього користувальницького інтерфейсу Windows, загортаючи існуючий API Windows в керований код. За допомогою Windows Forms, .NET Framework забезпечує більш комплексну абстракцію над API Win32, ніж це зробили Visual Basic або MFC.

Windows Forms подібний до бібліотеки Microsoft Foundation Class (MFC) при розробці клієнтських додатків. Він пропонує обгортку, що складається з набору класів C ++ для розробки програм Windows. Однак він не забезпечує рамки програми за замовчуванням, як MFC. Кожен елемент керування у програмі Windows Forms - це конкретний екземпляр класу.

Усі візуальні елементи в бібліотеці класів Windows Forms походять від класу Control. Це забезпечує мінімальну функціональність елемента

інтерфейсу користувача, такого як розташування, розмір, колір, шрифт, текст, а також загальні події, такі як натискання та перетягування/відпускання(drag/drop). Клас Control також має підтримку вміщення(docking), щоб дозволити елементу управління переставити своє положення під своїм батьківським. Підтримка Microsoft Active Accessibility в класі Control також допомагає користувачам з обмеженими можливостями краще використовувати Windows Forms.

Окрім надання доступу до рідних елементів керування Windows, таких як кнопка, текстове поле, прапорець та перегляд списку, Windows Forms додав власні елементи керування для хостингу ActiveX, розташування макетів, перевірки валідності та прив'язки великих даних. Ці елементи керуються за допомогою GDI +(Graphics Device Interface).

Як і абстрактний інструментарій віконних інструментів (AWT), еквівалентний API Java, Windows Forms був раннім і простим способом надання графічних компонентів інтерфейсу користувача до .NET Framework. Форми Windows побудовані на існуючому API Windows, а деякі елементи керування лише обробляють основні компоненти Windows. Деякі з методів дозволяють отримати прямий доступ до зворотних викликів Win32, які недоступні на платформах, що не належать до Windows.

У .NET Framework 2.0, Windows Forms отримав багатший контроль компонування, керування смужками інструментів у стилі Office 2003, багатопотокові компоненти, багатша підтримка дизайну та прив'язки даних, а також ClickOnce для веб-розгортання.

З випуском .NET 3.0 Microsoft випустила другий, паралельний API для візуалізації графічних інтерфейсів: Windows Presentation Foundation (WPF) на базі DirectX, а також декларативну мову GUI під назвою XAML.

Під час сеансу запитань і відповідей на конференції Build 2014 Microsoft пояснила, що Windows Forms перебуває в режимі технічного

обслуговування, без додавання нових функцій, але знайдені помилки все-таки виправлятимуться. Елементи керування формами були введені в оновленнях до .NET Framework версії 4.5.

Так як для розробки проекту використовується Windows Forms, то далі будуть наведені деякі використані елементи.

TableLayoutPanel. Представляє панель, яка динамічно викладає її вміст у сітку, що складається з рядків та стовпців.

Оскільки макет виконується як під час проектування, так і під час виконання, він може динамічно змінюватися, коли змінюється середовище програми. Це дає елементам управління на панелі можливість пропорційно змінювати розмір, тому він може реагувати на зміни, такі як розмір батьківського елемента управління або зміна довжини тексту через локалізацію.

Будь-який елемент керування формами Windows може бути дочірнім елементом керування TableLayoutPanel, включаючи інші екземпляри TableLayoutPanel. Це дозволяє побудувати складні макети, які адаптуються до змін під час виконання.

Елемент керування TableLayoutPanel може розширюватися для розміщення нових елементів керування, коли вони додаються, залежно від значення властивостей RowCount, ColumnCount та GrowStyle. Встановлення для властивості RowCount або ColumnCount значення 0 вказує, що TableLayoutPanel не буде пов'язаний у відповідному напрямку.

Ви також можете керувати напрямом розширення (горизонтальним чи вертикальним) після того, як керування TableLayoutPanel заповнене дочірніми елементами управління. За замовчуванням керування TableLayoutPanel розширюється вниз, додаючи рядки.

Якщо ви хочете, щоб рядки та стовпці поводилися інакше, ніж поведінка за замовчуванням, ви можете керувати властивостями рядків та

стовпців, використовуючи властивості RowStyles та ColumnStyles. Ви можете встановити властивості рядків або стовпців окремо.

Елемент керування TableLayoutPanel додає такі властивості дочірнім елементам управління: Cell, Column, Row, ColumnSpan та RowSpan.

Ви можете об'єднати комірки в контролі TableLayoutPanel, встановивши властивості ColumnSpan або RowSpan на дочірньому контролі.

Поведінка стикування дочірніх елементів керування така ж, як і інші елементи управління контейнерами.

Встановлення значень властивостей стовпців та рядків дочірнього керування до -1 призведе до переміщення елемента керування до першої порожньої комірки в керуванні TableLayoutPanel. Порожню клітинку буде обрано у пошуку, який ведеться зліва направо та зверху вниз. Цей порядок залежить від культури, тому він буде вести себе правильно в макетах справа-наліво (RTL).

Button. Елементи керування Windows Forms - це багаторазові компоненти, які інкапсулюють функціональний інтерфейс користувача та використовуються у клієнтських додатках Windows. Кнопка - це елемент керування, який представляє собою інтерактивний компонент, який дозволяє користувачам спілкуватися з додатком, який ми натискаємо та відпускаємо для виконання деяких дій.

RichTextBox. Керування RichTextBox дозволяє відображати або редагувати вміст потоку, включаючи абзаци, зображення, таблиці тощо.

І RichTextBox, і TextBox дозволяють користувачам редагувати текст, однак два елементи керування використовуються в різних сценаріях. RichTextBox - кращий вибір, коли користувачеві потрібно редагувати відформатований текст, зображення, таблиці чи інший багатий контент. Наприклад, редагування документа, статті чи блогу, що вимагає

форматування, зображень тощо, найкраще здійснювати за допомогою RichTextBox.

GroupBox. GroupBox відображає рамку навколо групи елементів керування з підписом або без нього. Використовуйте GroupBox, щоб логічно групувати набір елементів управління на формі.

Label. Елементи управління мітками зазвичай використовуються для надання описового тексту для елемента керування. Наприклад, ви можете використовувати мітку для додавання описового тексту для елемента TextBox, щоб повідомити користувача про тип даних, який очікується в елементі управління. Елементи керування мітками також можуть використовуватися для додавання описового тексту до Форми, щоб надати користувачеві корисну інформацію. Наприклад, ви можете додати мітку до верхньої частини форми, яка дає вказівки користувачеві про те, як вводити дані в елементи управління на формі. Елементи керування мітками також можуть використовуватися для відображення інформації про час роботи про стан програми.

MaskedTextBox. Клас MaskedTextBox - це вдосконалений елемент управління TextBox, який підтримує декларативний синтаксис для прийому або відхилення введення користувача. Використовуючи властивість Mask, ви можете вказати наступне введення, не записуючи у вашій програмі будь-якої спеціальної логіки перевірки:

- необхідні символи введення.
- необов’язкові символи введення.
- тип введення, що очікується в заданій позиції в масці; наприклад, цифра або алфавітний або буквено-цифровий символ.
- маскувальні літерали або символи, які повинні з'являтися безпосередньо в MaskedTextBox; наприклад, дефіси (-) в телефонному номері або символ валюти в ціні.

- спеціальна обробка символів введення; наприклад, перетворити алфавітні символи у великі регістри.

Клавішами зі стрілками можна використовувати для переходу до попереднього чи наступного положення.

Ви можете скористатися властивістю MaskFull, щоб перевірити, чи ввів користувач усі необхідні дані. Властивість Text завжди буде отримувати вхід користувача, відформатований відповідно до маски та властивості TextMaskFormat.

Елемент управління MaskedTextBox фактично відкладає всю обробку масок на клас System.ComponentModel.MaskedTextProvider, визначений властивістю MaskedTextProvider. Цей стандартний постачальник підтримує всі символи Unicode, крім сурогатів та вертикально комбінованих символів; однак властивість AsciiOnly може використовуватися для обмеження введення символів наборів a-z, A-Z та 0-9.

DataGridView. Елемент керування DataGridView забезпечує настроювану таблицю для відображення даних. Клас DataGridView дозволяє налаштувати комірки, рядки, стовпці та межі за допомогою використання властивостей, таких як DefaultCellStyle, ColumnHeadersDefaultCellStyle, CellBorderStyle та GridColor.

Ви можете використовувати елемент керування DataGridView для відображення даних із базовим джерелом даних або без нього. Не вказуючи джерело даних, ви можете створювати стовпці та рядки, що містять дані, і додавати їх безпосередньо до DataGridView, використовуючи властивості рядків та стовпців. Ви також можете використовувати колекцію рядків для доступу до об'єктів DataGridViewRow та властивості DataGridViewRow.Cells для прямого читання або запису значень комірок. Індексний елемент String, Int32] також забезпечує прямий доступ до комірок.

Як альтернатива заповненню елемента керування вручну, ви можете встановити властивості DataSource та DataMember так, щоб прив'язувати DataGridView до джерела даних та автоматично заповнювати його даними.

CheckedListBox. Цей елемент керування представляє перелік елементів, якими користувач може переходити за допомогою клавіатури або смуги прокрутки в правій частині елемента керування. Користувач може поставити галочку на один або декілька елементів.

Щоб додати об’єкти до списка під час виконання, призначте масив посилань на об’єкти методом AddRange. Потім у списка відображається значення рядка за замовчуванням для кожного об'єкта. Ви можете додати окремі елементи до списка методом Add.

Об'єкт CheckedListBox підтримує три стани через перерахунок CheckState: Checked, Indeterminate та Unchecked. Ви повинні встановити стан Indeterminate у коді, оскільки інтерфейс користувача для CheckedListBox не забезпечує механізм для цього.

2.3. Проектування графічного інтерфейсу

Для взаємодії користувача з програмою потрібний графічний інтерфейс. Він має бути простим, зручним, зрозумілим та відповідати вимогам програми. Для відображення мінімальної функціональності програмного продукта даної роботи потрібно мати хоча б поле введення учасників, а також область виведення. Проте для кращого функціоналу та майбутніх удосконалень додатку, були введені додаткові елементи інтерфейсу.

Все вікно програми містить елемент TableLayoutPanel, що представляє панель, яка динамічно вкладає її вміст у сітку, що складається з рядків та стовпців. Загалом TableLayoutPanel містить 3 рядки, кожний з яких має різне заповнення.

В першому рядку містяться 2 елементи RichTextBox для введеня даних команд(учасників), оп одному елементу для сіяних та несіяних гравців. Також тут розміщується елемент MaskedTextBox як поле введення кількості, в нашому випадку, кількості груп, що потрібно для розбиття учасників на групи(додатковий функціонал). Поряд з полем введеням кількості, розміщуються 2 кнопки(елемент Button).Одна з них для розподілу учасників по групам та побудови турнірної сітки, інша для очищення області виведення сітки учасників. Всі ці елементи зібрані за допомогою елементу GroupBox на 3 «області», кожна з яких має свої підписи(елемент label), для розуміння їх функціональності.

В другому ряді елемента TableLayoutPanel міститься область виведення відгрупованих учасників(елемент DataGridView). В цій області відображаються згруповані учасники(вертикально) в залежності від кількості введених груп.

В третьому ряді елемента TableLayoutPanel міститься область відображення турнірної сітки, де користувач вибирає переможця, що просуває турнірну таблицю.

Загалом інтерфейс має наступний вигляд, як зображено на рисунку 2.1.

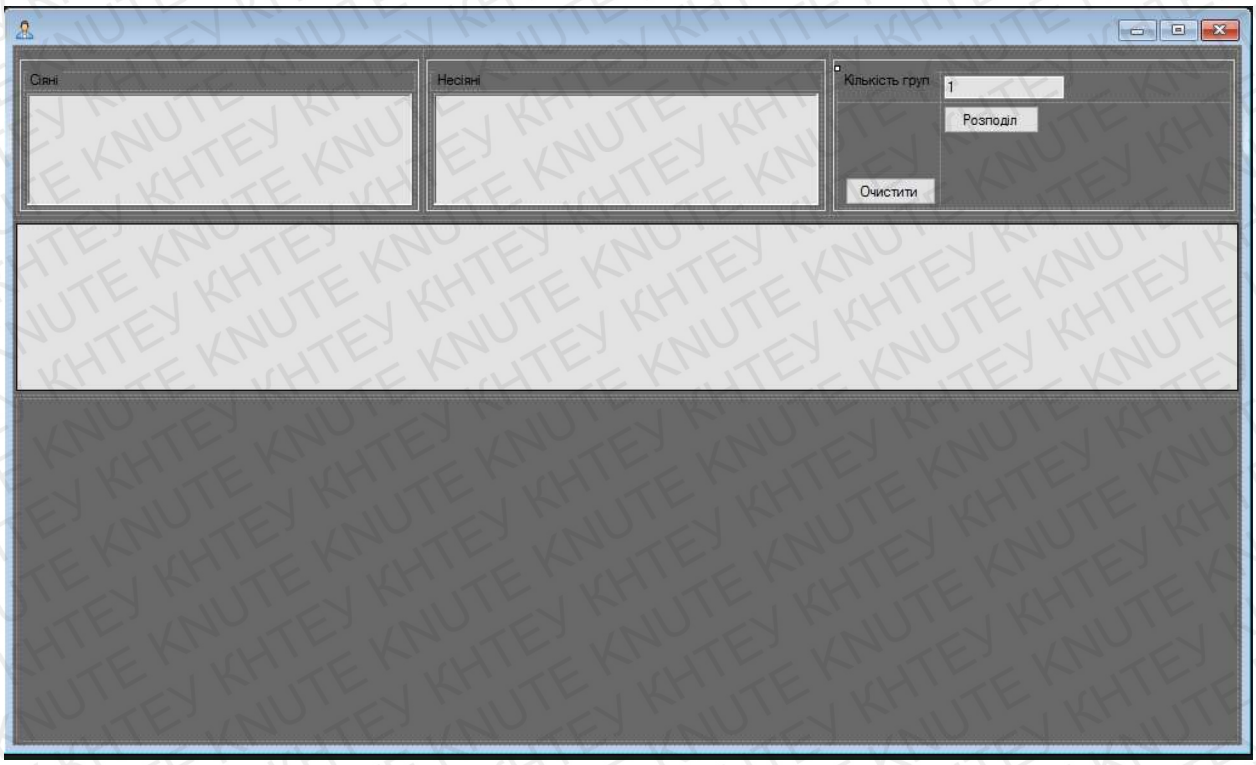

Рис. 2.1. Інтерфейс програми

2.4. Висновки до розділу 2

Під час тематичного дослідження було встановлено, що завданням роботи є розробка додатку «Жеребкування» для операційної системи Windows. Програмний продукт має мати функціонал для жеребкування, тобто розподілення команд по парам для подальшого визначення переможця. Для виконання даного завдання було обрано використовувати мову програмування C#. Розробка ведеться в Visual Studio за допомогою графічної бібліотеки Windows Forms.

В даному розділі було визначено основні вимоги до програмного продукта, а також розглянуто графічну бібліотеку WinForms та її основні елементи, що будуть використовуватися у роботі, а саме:

- *TableLayoutPanel.* Представляє панель, яка динамічно викладає її вміст у сітку, що складається з рядків та стовпців.

- *Button.* Елемент керування, який представляє собою інтерактивний компонент, який дозволяє користувачам спілкуватися з додатком, який ми натискаємо та відпускаємо для виконання деяких дій.
- *RichTextBox.* Керування RichTextBox дозволяє відображати або редагувати вміст потоку, включаючи абзаци, зображення, таблиці тощо.
- *Label.* Елементи управління мітками зазвичай використовуються для надання описового тексту для елемента керування.
- *MaskedTextBox. Ц*е вдосконалений елемент управління TextBox, який підтримує декларативний синтаксис для прийому або відхилення введення користувача.
- *DataGridView.* Елемент керування DataGridView забезпечує настроювану таблицю для відображення даних.
- *CheckedListBox.* Цей елемент керування представляє перелік елементів, якими користувач може переходити за допомогою клавіатури або смуги прокрутки в правій частині елемента керування. Користувач може поставити галочку на один або декілька елементів.

Також було спроектовано графічний користувацький інтерфейс(дивись *рис. 2.1*) в якому використовуються вищевказані елементи WinForms.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ПРОДУКТА

3.1. Структура додатку

Загалом проект складається з 3 класів, кожен з яких виконує свою функцію:

- Клас Program – головна точка входу програми, він запускає головну форму(вікно програми);
- Клас Form1(Form1.Designer) – цей клас складається з 2 файлів, він так званий partial class(частковий клас). У файлі Form1.Designer.cs ініціалізуються та задаються параметри для елементів графічного інтерфейсу. У файлі Form1.cs, головному вікні програми, містяться функції для обробки даних, ініціалізації динамічних елементів, управління та взаємозвязку інтерфейсу.
- Клас RandomStringArrayTool – відповідає за алгоритм «сортування» масиву, іншими словами, клас містить функції для переміщення елементів масиву даних у випадковому порядку.

На рисунку 3.1 відображена UML-діаграмма класів проекту. Так як клас Form1 є «частковим класом», то його елементи відображені в 2 файлах, на діаграмі відображено сірим елементи файлу Form1.Designer.cs.

Головна форма містить декілька методів:

- b_init_Click – метод, що запускається при натисненні кнопки «Розподіл»;
- output – виводить масив учасників на область, що відображає розбиття на групи;
- output2 – створює турнірну сітку учасників та заповнює її, складається з елементів CheckedListBox;
- AddButton – створює кнопку знизку сітки учасників;
- NextButton_Click – метод, що запускається при натисненні кнопки «Далі»;
- getPlayers – отримання списка учасників з поля введення;
- Clear_Click- метод, що запускається при натисненні кнопки «Очистити»;
- Explode – розбиває рядок на підрядок.

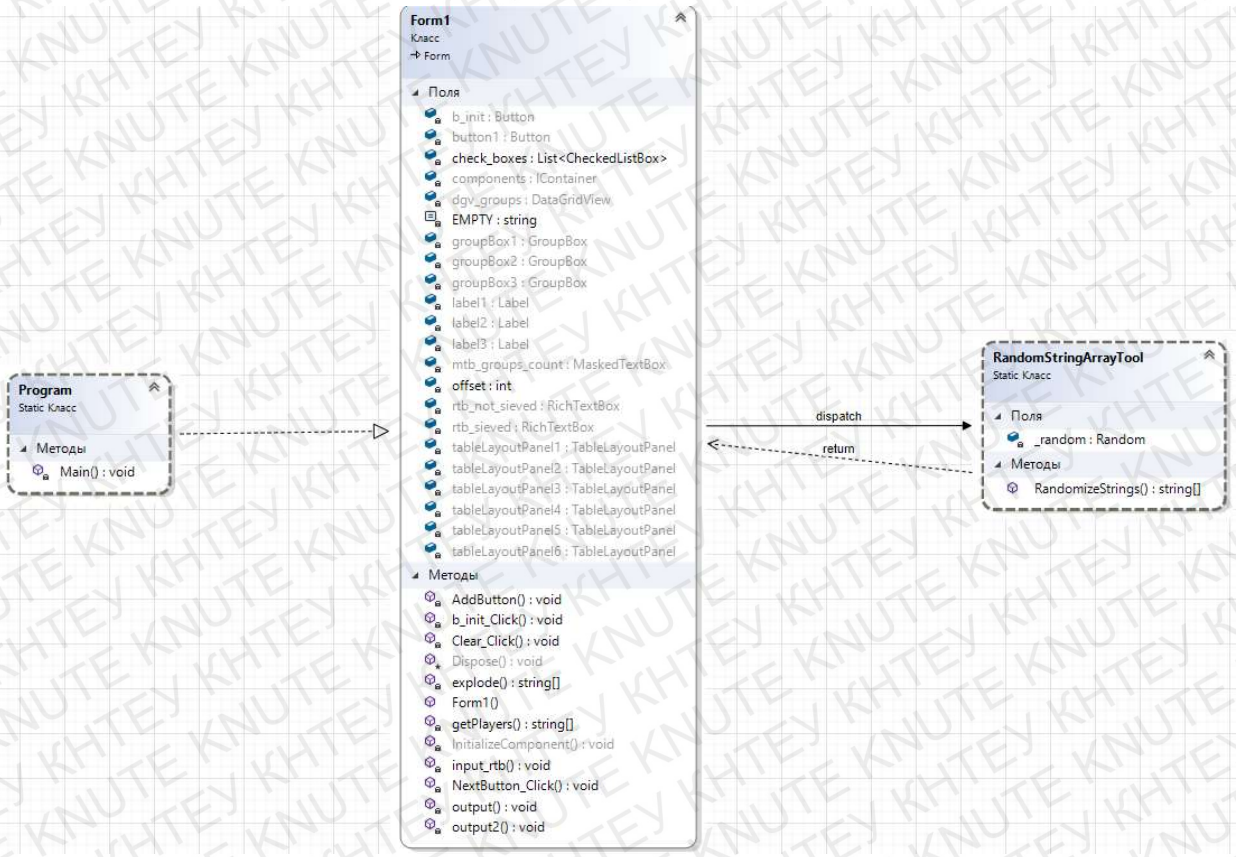

Рис. 3.1. UML-діаграмма класів

3.2. Послідовність розробки

Загалом нас цікавлять два класи – це Form1 та RandomStringArrayTool. Розглянемо спочатку клас Form1.

Так як ми розроблюємо програмний продукт за допомогою WinForms, то зручно буде спочаку створити основні елементи інтерфейсу. Для цього в файлі Form1.Designer.cs(частина класу яка відповідає за ініціалізацію об’єктів та їх параметри) ініціалізуємо необхідні елементи, що вже були встановлені в розділі проектування графічного інтерфейсу(див. розділ 2.3). На рисунку 3.1 ми задаємо основні елементи, що були відображені на рисунку 2.1.

```
this.rtb_sieved = new System.Windows.Forms.RichTextBox();
this.b_init = new System.Windows.Forms.Button();
this.rtb_not_sieved = new System.Windows.Forms.RichTextBox();
this.tableLayoutPanel1 = new System.Windows.Forms.TableLayoutPanel();
this.groupBox1 = new System.Windows.Forms.GroupBox();
this.tableLayoutPanel2 = new System.Windows.Forms.TableLayoutPanel();
this.label1 = new System.Windows.Forms.Label();
this.groupBox2 = new System.Windows.Forms.GroupBox();
this.tableLayoutPanel3 = new System.Windows.Forms.TableLayoutPanel();
this.label2 = new System.Windows.Forms.Label();
this.groupBox3 = new System.Windows.Forms.GroupBox();
this.tableLayoutPanel5 = new System.Windows.Forms.TableLayoutPanel();
this.label3 = new System.Windows.Forms.Label();
this.mtb_groups_count = new System.Windows.Forms.MaskedTextBox();
this.button1 = new System.Windows.Forms.Button();
this.tableLayoutPanel4 = new System.Windows.Forms.TableLayoutPanel();
this.dgv_groups = new System.Windows.Forms.DataGridView();
this.tableLayoutPanel6 = new System.Windows.Forms.TableLayoutPanel();
```

Рис.3.1. Ініціалізація графічних елементів

Далі задаємо кожному статичному елементу необхідні параметри. На рис. 3.2 ми бачимо параметри елемента tableLayoutPanel6, де наприклад рядок «this.tableLayoutPanel6.Dock = System.Windows.Forms.DockStyle.Fill;» задає елементу заповнити батьківський елемент(tableLayoutPanel4), від краю до краю. Детальні параметри можете переглянути в лістингу коду.

```
// tableLayoutPanel6
//
this.tableLayoutPanel6.AutoScroll = true;
this.tableLayoutPanel6.AutoSizeMode = System.Windows.Forms.AutoSizeMode.GrowAndShrink;
this.tableLayoutPanel6.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 0F));
this.tableLayoutPanel6.Dock = System.Windows.Forms.DockStyle.Fill;
this.tableLayoutPanel6.GrowStyle = System.Windows.Forms.TableLayoutPanelGrowStyle.AddColumns;
this.tableLayoutPanel6.Location = new System.Drawing.Point(3, 291);
this.tableLayoutPanel6.Name = "tableLayoutPanel6";
this.tableLayoutPanel6.RowStyles.Add(new System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 0F));
this.tableLayoutPanel6.Size = new System.Drawing.Size(1007, 283);
this.tableLayoutPanel6.TabIndex = 12;
//
```

Рис 3.2. Параметри елемента tableLayoutPanel6

Далі переходимо до файлу Form1.cs, як вже згадувалося в ньому є декілька методів, розглянемо їх детальніше. Для кращого розуміння програми продемонструємо роботу додатку з фрагментами коду. При відкритті програма має вигляд спроектованого інтерфейса*(рис. 2.1)*. Введемо

деякі дані в поля введення, в нашому випадку «гравцями» будуть цифри від 1 до 8(*рис.3.3*).

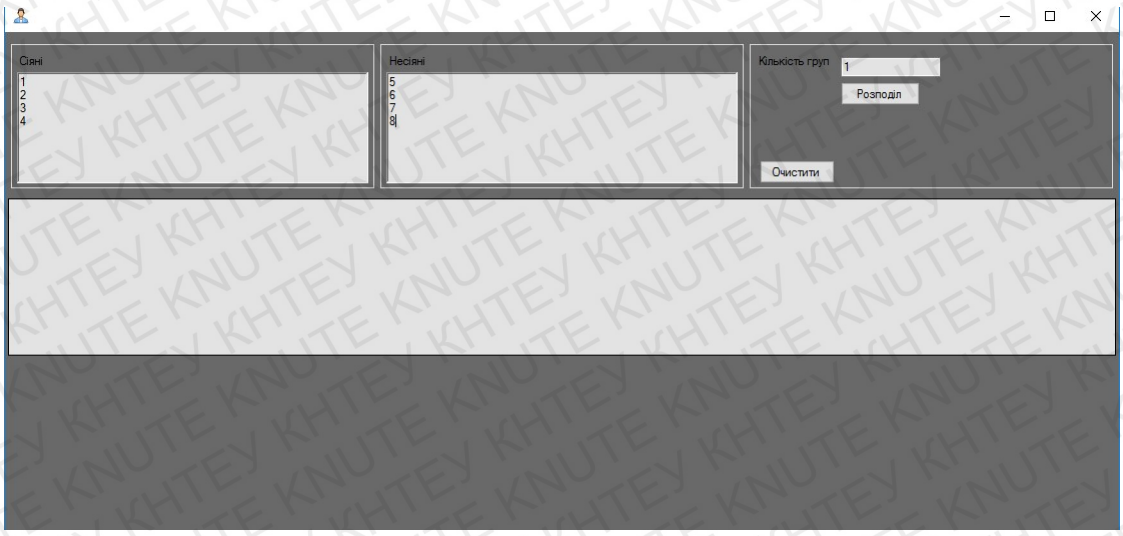

Рис. 3.3. Введення початкових гравців

Після натиснення кнопки «Розподіл», без зміни кількості груп, програма виведе 1 групу гравців в перше поле виведення, та 4 CheckedListBox, що формують пари, в друге поле виведення*(рис.3.4)*.

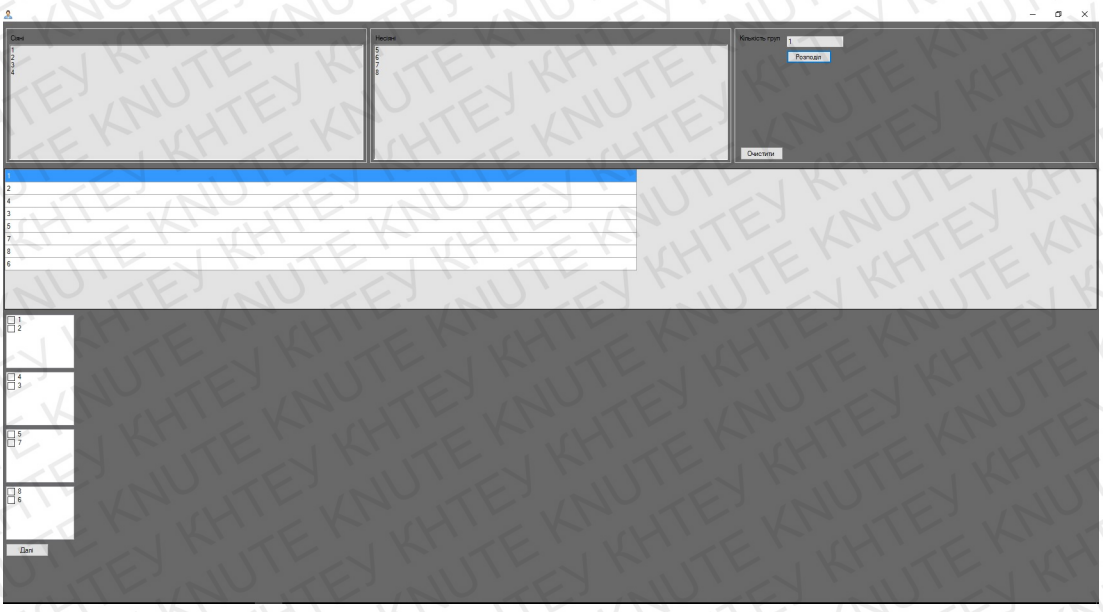

Рис.3.4. Початковий розподіл

При зміні кількості груп та натисненні кнопки «Розподіл» другий розділ відповідно відобразить задану кількість групп*(рис.3.5)*, зазначимо, що при кожному натисненні кнопки «Розподіл», дані турнірної сітки не зберігаються й будуть перерозподілені.

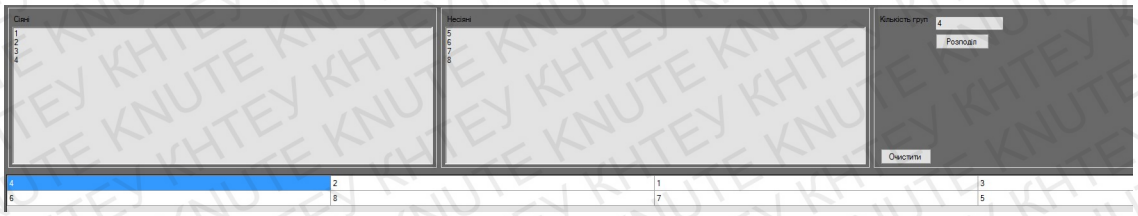

Рис.3.5. Розбиття учасників на 4 групи

Повернемося до коду програми, під час цих дій було задіяно декілька методів. Перший виклик має метод b_init_Click, який виконується при натисненні кнопки «Розподіл». Метод створює масив рядків, елементами якого є введені в поля гравці, для цього викликається метод getPlayers, а потім метод RandomizeStrings з класу RandomStringArrayTool для перемішення учасників*(рис.3.6)*. Після чого викликаються: метод output – для відображення груп учасників, метод output2 – для «створення» турнірної сітки, та метод AddButton – для створення(ініціалізації) кнопки знизу сітки.

```
private void b_init_Click(object sender, EventArgs e)
{
    offset = 0;
    string[] sieved = getPlayers(rtb_sieved);
    string[] not_sieved = getPlayers(rtb_not_sieved);
   not_sieved = RandomStringArrayTool.RandomizeStrings(not_sieved);
   sieved = RandomStringArrayTool.RandomizeStrings(sieved);
```

Рис.3.6. Фрагмент коду методу b_init_Click

Метод output2 має велике значення для програми, він ініціалізує новий екземпляр CheckedListBox, визначає місце розміщення та заповнює двома гравцями(рис 3.7). Кожний раз список екземплярів очищується, а позиція перераховуються, що дозволя методу використовуватися декілька разів, адаптимно до кількості введених учасників.

```
119     private void output2(TableLayoutPanel tableLayout, string[] ar, int offset, int mc_c)
120     {
121         /* Column Index , Row index */
122          int number = 0;
123         for (int j = 0; j < (ar.Length/2); j++) //вывод боксов
124           {// добавляем чеклист
125            CheckedListBox newBox = new CheckedListBox();
126            check_boxes.Add(newBox);
127            newBox.Name = "newBox" + j;
128            newBox.CheckOnClick = true;
129            tableLayout.Controls.Add(newBox, offset, j);
130            // заполнение боксов
131            if (ar[number] != null)  newBox.Items.Add(ar[number]); number++;
132            if (ar[number] != null)  newBox.Items.Add(ar[number]); number++;
133           }
134     }
```

Рис. 3.7. Лістинг коду методу output2

Після виконання методу output2, будується кнопка «Далі» за допомогою метода AddButton, при натисненні вона викликає метод NextButton_Click. Але спочатку вернемося до інтерфейсу програми та виберемо учасників котрі перемогли за допомогою кліку на них(поряд відобразиться галочка, а учасник буде вибраний синім кольором)*(рис.3.8)*.

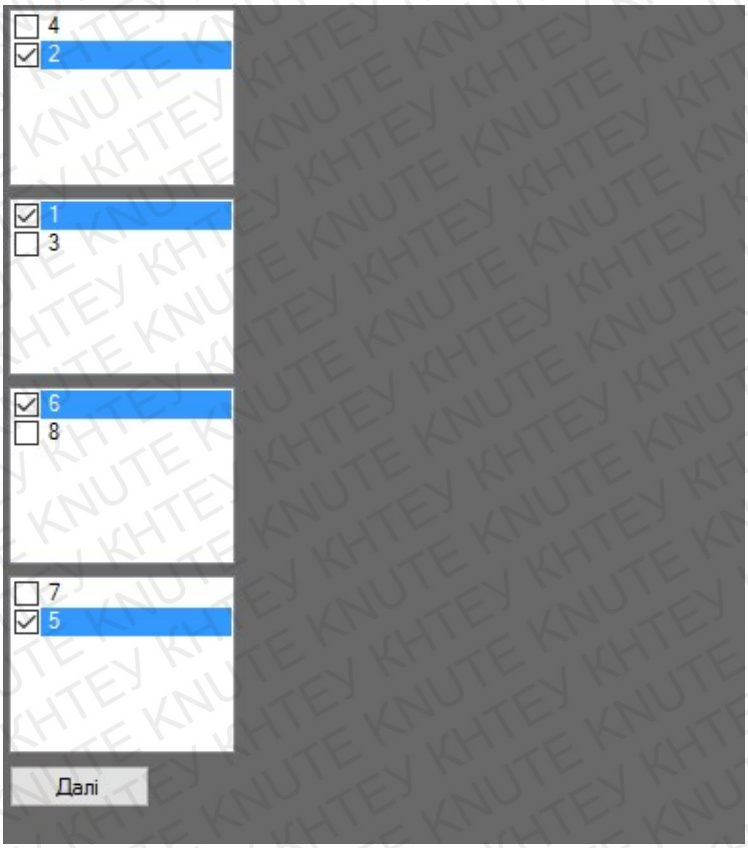

Рис.3.8. Вибір переможців першого етапу

При натисненні на кнопку «Далі» «будується» наступний етап турніру, тут варто зазначити, що можна вибрати 2 учасників одразу, що буде означати нічью. При натисненні кнопки спрацьовує метод NextButton_Click(*рис.3.9*), який перевіряє екземпляри CheckedListBox на вибір учасників(переможців), підраховує їх кількість та створює відповідний масив з переможцями, далі масив випадково перемішується, й знову викликаються методи ouput2 та AddButton, доки не залишиться єдиний переможець.

```
153         private void NextButton_Click(object sender, EventArgs e)
154         {
155                     int q = 0; //считает позицию вставки
156             int k = 0;//кол-во выбраных елементов
157             //считаем количество выбраных елементов
158             for (int i = 0; i < check_boxes.Count; i++)
159             {
160                 if (check_boxes[i].CheckedItems.Count != 0)
161                 {
162                     k += check_boxes[i].CheckedItems.Count;
163                 }
164             }
165             if (k > 0) offset++; else
166             { MessageBox.Show("Данні відсутні"); return; };
167             if (k % 2 !=0) k++;
168             string[] s = new string[k];
169             for (int i = 0; i < check_boxes.Count; i++)
170             {
171                 if (check_boxes[i].CheckedItems.Count != 0)
172                     {
173
174                     string[] x = new string[check_boxes[i].CheckedItems.Count];
175                     for (int j = 0; j < check_boxes[i].CheckedItems.Count; j++)
176                     {
177                         x[j] = check_boxes[i].CheckedItems[j].ToString();
178                     }
179                     Array.Copy(x, 0, s, q, x.Length);
180                     q += x.Length;
181                      }
182             }
183             s = RandomStringArrayTool.RandomizeStrings(s);
184             if (s.Length == 2 && s[0] == null || s[1] == null)
185             {
186                 MessageBox.Show("Команда '" + s[0] + s[1] + "' перемогла!"); return;
187             }
188             check_boxes.Clear();
189             output2(tableLayoutPanel6, s, offset, s.Length);
190             AddButton(tableLayoutPanel6, s, offset, s.Length);
191         }
```

Рис.3.9. Лістинг коду методу NextButton_Click

При проходженні всіх етапів турніру, коли виявлено переможця, виводиться відповідне повідомлення(рис.3.10).

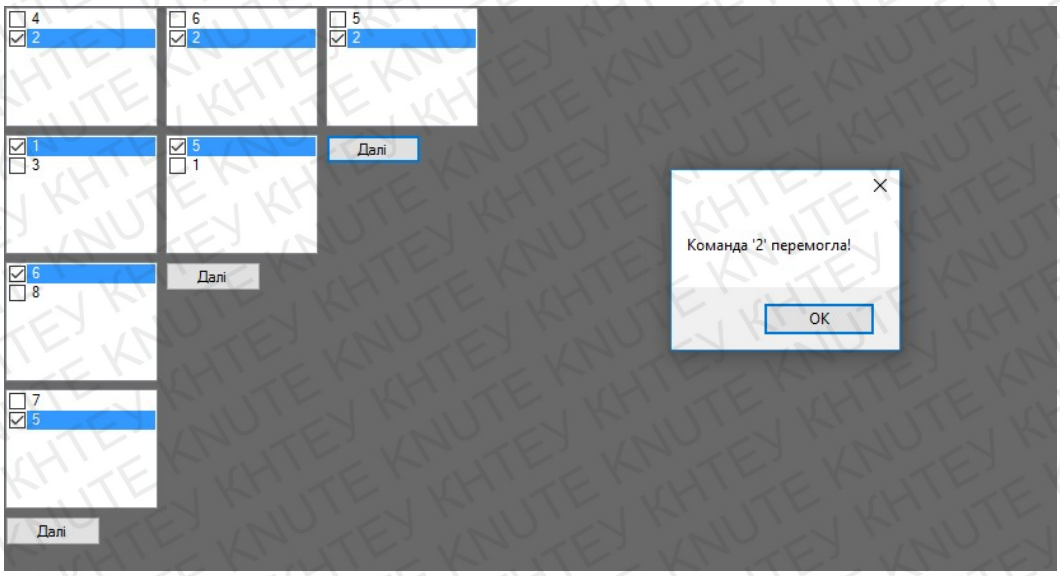

Рис.3.10. Повідомлення про переможця

3.3. Висновки до розділу 3

Під час розробки програмного забезпечення важливе місце займає вибір середовища розробки. В даній роботі використовувався Visual Studio, так як ця середа розробки зручна в використанні та може працювати з Windows Forms.

Під час роботи над даною роботою було встановлено структуру програмного продукта, а також побудована UML-діаграмма класів, на якій відображено взаємодія між класами.

Під час виконання роботи було створено 3 класи, кожен з яких виконує свою функцію:

- Клас Program – головна точка входу програми, ініціалізує головну форму(вікно програми);
- Клас Form1(Form1.Designer) – цей клас складається з 2 файлів, він так званий partial class(частковий клас). У файлі Form1.Designer.cs ініціалізуються та задаються параметри для елементів графічного інтерфейсу. У файлі Form1.cs, головному

вікні програми, містяться функції для обробки даних, ініціалізації динамічних елементів, управління та взаємозвязку інтерфейсу.

- Клас RandomStringArrayTool – відповідає за алгоритм «сортування» масиву, іншими словами, клас містить функції для переміщення елементів масиву даних у заданому (випадковому) порядку.

В даному розділі було розглянуто послідовність створення програмного продукту «Жеребкування» для операційної системи Windows з використанням графічного інтерфейсу WinForms на мові програмування C#. Також було продемонстровано взаємозв’язок між елементами графічного інтерфейсу та коду програми, й було пояснено деякі частини розробки даного програмного забезпечення.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

У ході виконання дипломної роботи були більш докладно вивчені і викладені наступні питання:

- аналітичний огляд предметної області;
- написання технічного завдання програмного продукта;
- робота з графічною бібліотекою Windows Forms
- розгляд створення програмного забезпечення з використанням Windows Forms для операційної системи Windows;
- розробка додатку "Жеребкування" на платформі Windows;
- тестування створеного додатку.

Під час роботи було створено додаток «Жеребкування», за допомогою якого користувачі можуть групувати сіяних та несіяних учасників та створювати турнірну сітку за методом олімпійської системи (Single Elimination). Проте слід зауважити, що в стандартній олімпійській системі, кількість учасників має бути ступенем двійки, в данній програмі такого обмеження немає. Відповідно до правил системи з одною поразкою, останній гравець перемагає, а програвший йому отримує друге місце.

Програмний продукт, що був створенний у ході виконання дипломної роботи може бути використаний на кіберспортивних турнірах КНТЕУ.

Додаток «Жеребкування» є логічно завершеним програмним продуктом. Проте у майбутньому можливі зміни і збільшення функціоналу, наприклад збільшення кількості алгоритмів сортування, чи створення турнірної сітки по групам, або ж просто оновлення графічного інтерфейсу.

Додаток був створений таким чином, щоб його оновлення було легко проводити.

Данний програмний продукт має зацікавити організаторів кіберспортивних змагань, так як має легкий, інтуїтивний інтерфейс та необхідний функціонал.

.

Список використаних джерел

Література

1. Агуров, Павел «C#. Сборник рецептов» / Павел Агуров. - М.: "БХВ-Петербург", 2012. - 432 с
2. Вагнер, Билл «C# Эффективное программирование» / Билл Вагнер. - М.: ЛОРИ, 2013. - 320 с.
3. Касаткин, А. И. «Профессиональное программирование на языке си. Управление ресурсами» / А.И. Касаткин. - М.: Высшая школа, 2012. - 432 с.
4. Рихтер, Джеффри «CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C#» / Джеффри Рихтер. - М.: Питер, 2013. - 928 с.
5. Захист додатків від злому: журнал Безпека соціально-економічних процесів в кіберпросторі / Сперисенко О.В., Пашорін В.І. – К.: Київ. нац. торг.-екон. ун-т, 2019 – 119-120 с.
6. Ишкова, Э. А. «Самоучитель C#. Начала программирования» / Э.А. Ишкова. - М.: Наука и техника, 2013. - 496 с.

Інтернет-ресурси

7. Microsoft – Docs [Электронный ресурс]. –Режим доступа: https://docs.microsoft.com/en-us/dotnet/framework/winforms/
8. Newzoo Analytics [Электронный ресурс]. –Режим доступа: https://newzoo.com/

					КНТЕУ 121– 05-19.МР		
					Розробка програмного забезпечення для кіберспортивних змагань	*Стадія*	*Аркуш*	*Аркушів*
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*		СВД	*48*	*48*
Зав. кафедри		Криворучко О.В.						
Керівник		Пашорін В.І.						
Гарант		Криворучко О.В.			*Список використаних джерел*	Факультет обліку, аудиту та інформаційних систем, 2м курс, 5 група		
Розроб.		Сперисенко О.В.						

ДОДАТКИ

Программа та методика тестування

Для тестування даного проекту, була обрана методика «Manual testing», тобто метод ручного тестування. Цей спосіб є менш затратим та потребує менше навиків, проте дає хороші результати.

Для тестування додатку, були використанні наступні види тестування:

1. Функціональне тестування.

   Було протестовано функції додатку, щоб бути впевними, що вони працюють коректно.

2. Дослідницьке тестування.

   Було запрошено потенційного користувача з цільової аудиторії, без досвіду тестування, якій надали версію проекту, для того, що б перевірити, чи зрозумілий дизайн та функціонал додатку.

3. Регресивне тестування.

   Після того, як відбувалися глобальні зміни в коді додатку, проводилося повторне тестування всіх функцій та компонентів, щоб забезпечитися в тому, що після змін та виправленя багів, весь функціонал працює коректно.

Для подальшого полегшення тестування були розроблені та використані декілька тест кейсів.

Тест кейс має наступні складові:

1. Порядковий номер тест кейсу
2. Описання тестового кейсу
3. Кроки виконання
4. Отриманний результат
5. Очікуванний результат

Тест кейси

1. ТК_01_start
2. Перевірка завантаження додатку на операційній системі Windows (10)
3. Кроки виконання:
   3.1. Завантажити програму, шляхом подвійного кліку
   3.2. Спробувати змінити розміри вікна
   3.3. Ввести декілька учасників у поля учасників(сіяні, несіяні)
   3.4. Змінити кількість груп з 1 до 9 (довільно)
4. Додаток завантажився, розміри вікна змінюються, поля для введення працюють.
5. Додаток завантажується, розміри вікна змінюються, поля для введення записують дані.

1. ТК_02_work
2. Перевірка роботи додатку на операційній системі Windows (10)
3. Кроки виконання:
   3.1. Завантажити програму, шляхом подвійного кліку
   3.2. Ввести декілька учасників у поля учасників(сіяні, несіяні)
   3.3. Змінити кількість груп з 1 до 9 (довільно)
   3.4. Натиснути клавішу «Розподіл»
   3.5. Натиснути клавішу «Очистити»
4. Додаток завантажився, поля для введення працюють, учасники групуються відповідно до введеної кількості, будується початкова турнірна сітка(розбиття учасників по парам), знизу відображається кнопка «Далі», після натискання кнопки «Очистити» турнірна сітка зникає.
5. Додаток завантажився, поля для введення працюють, учасники групуються відповідно до введеної кількості, будується початкова турнірна сітка(розбиття учасників по парам), знизу відображається кнопка «Далі», після натискання кнопки «Очистити» турнірна сітка зникає.

1. ТК_03
2. Перевірка формування та роботи турнірної сітки
3. Кроки виконання:

   3.1. Завантажити програму, шляхом подвійного кліку

   3.2. Ввести декілька учасників у поля учасників(сіяні, несіяні)

   3.3. Натиснути клавішу «Розподіл»

   3.4. З пар учасників вибрати переможців, шляхом кліку по ним

   3.5. Натиснути кнопку «Далі»

   3.6. Повторювати пункти 3.4 та 3.5 доки не залишиться 2 учасника

   3.7. Натиснути на 1 з двох учасників та натиснути «Далі»
4. Додаток завантажився, поля для введення працюють, будується початкова турнірна сітка(розбиття учасників по парам), знизу відображається кнопка «Далі»; при клікі на учасника(з пари), він виділяється синім а зліва формується галочка, після натискання «Далі» будуються наступні пари з обраних учасників; при виборі останнього учасника з’являється вікно з надписом «Команда * перемогла».
5. Додаток завантажився, поля для введення працюють, будується початкова турнірна сітка(розбиття учасників по парам), знизу відображається кнопка «Далі»; при клікі на учасника(з пари), він виділяється синім а зліва формується галочка, після натискання «Далі» будуються наступні пари з обраних учасників; при виборі останнього учасника з’являється вікно з надписом «Команда * перемогла».

Керівництво користувача

Після запуску програми буде відображено головне вікно програми*(рис.1)*.

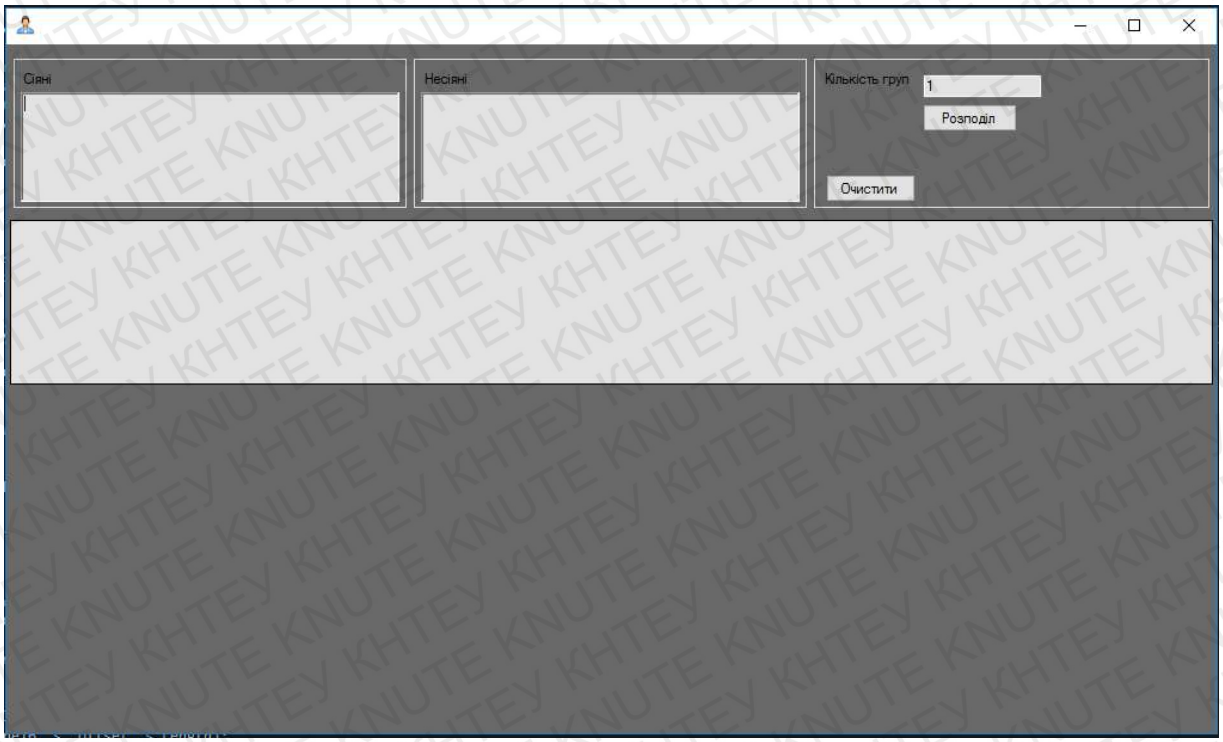

Рис 1. Головне вікно програми

Для початку роботи потрібно ввести учасників в поля «Сіяні» та «Несіяні». Введення відбувається за допомогою клавіатури, перехід до наступного учасника відбувається натисненням клавіші Enter. Після введення потрібних учасників, їх в будь-який час можна редагувати.

По необхідності ввести в поле «Кількість груп» необхідну кількість груп, для розбиття учасників на дану кількість груп. В дане поле можливо вводити лише цифри. Максимальна можлива кількість груп становить 9.

Після заповнення необхідних даних потрібно натиснути кнопку «Розподіл» в правій верхній частині вікна.

Для прикладу введемо учасників від 1 до 8, кількістю груп 4 та запустимо жеребкування шляхом натиснення кнопки «Розподіл». Матимемо наступне вікно(вікно може не вміщувати весь контент при стандартному розмірі, при необхідності його слід розтягнути або відкрити на весь екран), що зображено на рисунку 2.

Як бачимо, під областю введення учасників маємо область виведення згрупованих учасників(група йде вертикально). Нижче, учасники розбиті по парам у окремих блоках. Для визначення переможця пари, потрібно клікнути на учасника, зліва відобразиться галочка, а сам учасник буде виділений синім. Можливо не вибирати учасників якоїсь пари(в разі виключення), й вони не будуть брати участь у наступному етапі. При розіграші нічиї достатньо вибрати обох учасників.

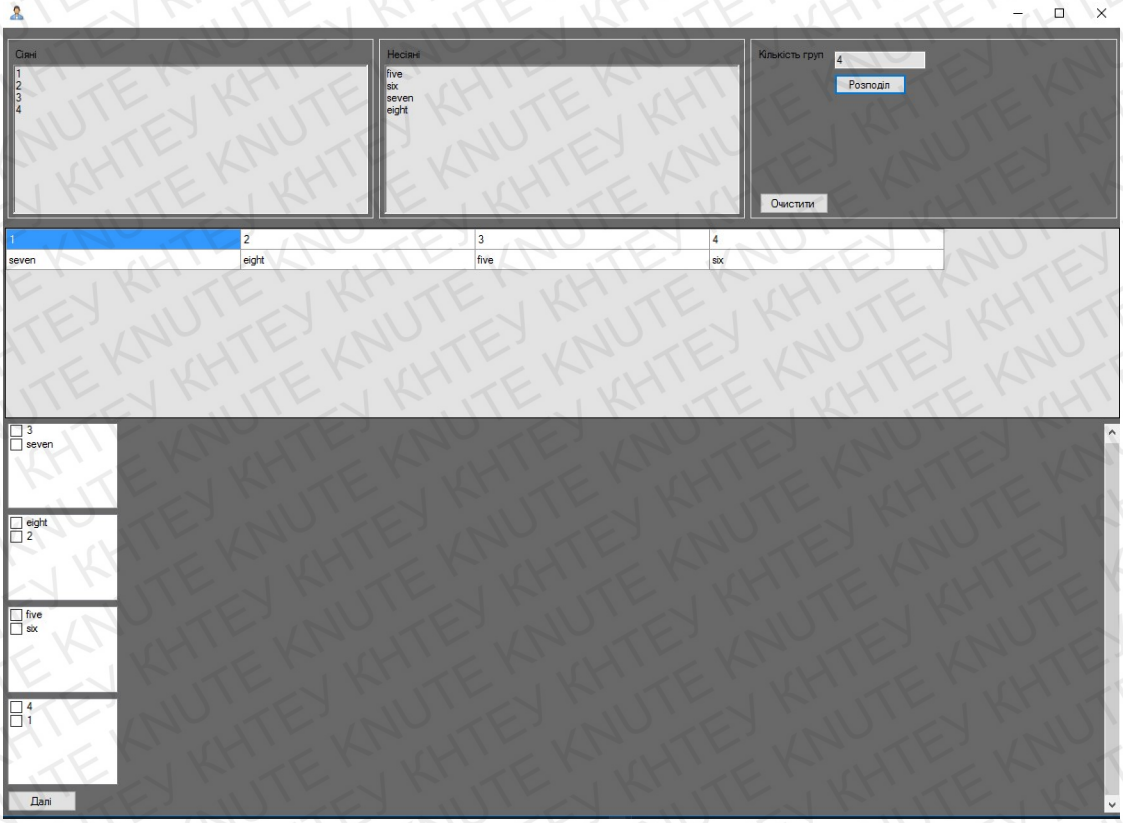

Рис. 2. Початкове сортування

Після визначення переможців/переможених натискаємо кнопку «Далі» знизу пар учасників. Отримуємо наступний етап з парами учасників. Повторює попередні дії, доки не буде визначено єдиний переможець, про що буде сповіщено повідомленням*(рис.3)*.

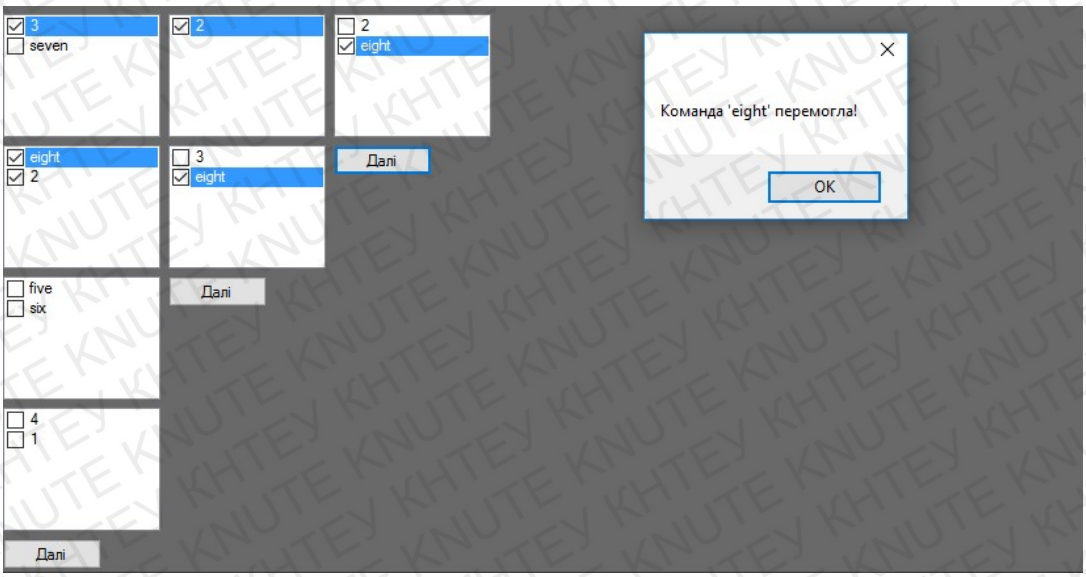

Рис. 3. Повідомлення про переможця турніру

Щоб розпочати нове жеребкування достатньо ввести нових учасників та натиснути «Розподіл», для очищення області виведення пар необхідно натиснути кнопку «Очистити», що поряд з кнопкою «Розподіл». Проте будьте уважні, результати(переможців) змінити не можливо, в випадки помилки потрібно розпочинати спочатку.

Лістинг коду

Додаток А

Клас Program

```
/// точка входу додатку
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{   static class Program
   {            [STAThread]
      static void Main()
      {          Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new Form1());       }   }}
```

Додаток Б

Клас RandomStringArrayTool

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace WindowsFormsApplication1
{   static class RandomStringArrayTool
   {            /// Stores the current random number
      static Random _random = new Random();
      /// Return randomized version of the byte array
      public static string[] RandomizeStrings(string[] arr)
```

```
{       List<KeyValuePair<int, string>> list = new List<KeyValuePair<int, string>>();
    // Add all bytes from array
    // Add new random int each time
    foreach (string s in arr)       {
      list.Add(new KeyValuePair<int, string>(_random.Next(), s));       }
    // Sort the list by the random number
    var sorted = from item in list
                 orderby item.Key
                 select item;
    // Allocate new byte array
    string[] result = new string[arr.Length];
    // Copy values to array
    int index = 0;
    foreach (KeyValuePair<int, string> pair in sorted)
    {           result[index] = pair.Value;
      index++;       }
    // Return copied array
    return result;      }   }}
```

Додаток В

Клас Form1(Form1.Designer.cs)

```
namespace WindowsFormsApplication1
{   partial class Form1
  {     /// <summary>
    /// Required designer variable.
   /// </summary>
```

```
        private System.ComponentModel.IContainer components = null;

        /// <summary>

        /// Clean up any resources being used.

        /// </summary>

        /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

        protected override void Dispose(bool disposing)

        {           if (disposing && (components != null))

           {               components.Dispose();

           }           base.Dispose(disposing);       }

        #region Windows Form Designer generated code


        /// <summary>

        /// Required method for Designer support - do not modify

        /// the contents of this method with the code editor.

        /// </summary>

        private void InitializeComponent()

        {

           System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(Form1));

           this.rtb_sieved = new System.Windows.Forms.RichTextBox();

           this.b_init = new System.Windows.Forms.Button();

           this.rtb_not_sieved = new System.Windows.Forms.RichTextBox();

           this.tableLayoutPanel1 = new System.Windows.Forms.TableLayoutPanel();

           this.groupBox1 = new System.Windows.Forms.GroupBox();

           this.tableLayoutPanel2 = new System.Windows.Forms.TableLayoutPanel();

           this.label1 = new System.Windows.Forms.Label();

           this.groupBox2 = new System.Windows.Forms.GroupBox();
```

```
this.tableLayoutPanel3 = new System.Windows.Forms.TableLayoutPanel();
this.label2 = new System.Windows.Forms.Label();
this.groupBox3 = new System.Windows.Forms.GroupBox();
this.tableLayoutPanel5 = new System.Windows.Forms.TableLayoutPanel();
this.label3 = new System.Windows.Forms.Label();
this.mtb_groups_count = new System.Windows.Forms.MaskedTextBox();
this.button1 = new System.Windows.Forms.Button();
this.tableLayoutPanel4 = new System.Windows.Forms.TableLayoutPanel();
this.dgv_groups = new System.Windows.Forms.DataGridView();
this.tableLayoutPanel6 = new System.Windows.Forms.TableLayoutPanel();
this.tableLayoutPanel1.SuspendLayout();
this.groupBox1.SuspendLayout();
this.tableLayoutPanel2.SuspendLayout();
this.groupBox2.SuspendLayout();
this.tableLayoutPanel3.SuspendLayout();
this.groupBox3.SuspendLayout();
this.tableLayoutPanel5.SuspendLayout();
this.tableLayoutPanel4.SuspendLayout();
((System.ComponentModel.ISupportInitialize)(this.dgv_groups)).BeginInit();
this.SuspendLayout();
//
// rtb_sieved
//
this.rtb_sieved.BackColor = System.Drawing.SystemColors.ControlLight;
this.rtb_sieved.Dock = System.Windows.Forms.DockStyle.Fill;
this.rtb_sieved.Location = new System.Drawing.Point(3, 18);
```

```
this.rtb_sieved.Name = "rtb_sieved";
this.rtb_sieved.Size = new System.Drawing.Size(317, 92);
this.rtb_sieved.TabIndex = 0;
this.rtb_sieved.Text = "";
//
// b_init
//
this.b_init.Location = new System.Drawing.Point(88, 28);
this.b_init.Name = "b_init";
this.b_init.Size = new System.Drawing.Size(79, 23);
this.b_init.TabIndex = 8;
this.b_init.Text = "Розподіл";
this.b_init.UseVisualStyleBackColor = true;
this.b_init.Click += new System.EventHandler(this.b_init_Click);
//
// rtb_not_sieved
//
this.rtb_not_sieved.BackColor = System.Drawing.SystemColors.ControlLight;
this.rtb_not_sieved.Dock = System.Windows.Forms.DockStyle.Fill;
this.rtb_not_sieved.Location = new System.Drawing.Point(3, 18);
this.rtb_not_sieved.Name = "rtb_not_sieved";
this.rtb_not_sieved.Size = new System.Drawing.Size(317, 92);
this.rtb_not_sieved.TabIndex = 9;
this.rtb_not_sieved.Text = "";
//
// tableLayoutPanel1
```

```
            //
            this.tableLayoutPanel1.ColumnCount = 3;
            this.tableLayoutPanel1.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 33.33333F));
            this.tableLayoutPanel1.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 33.33333F));
            this.tableLayoutPanel1.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 33.33333F));
            this.tableLayoutPanel1.Controls.Add(this.groupBox1, 0, 0);
            this.tableLayoutPanel1.Controls.Add(this.groupBox2, 1, 0);
            this.tableLayoutPanel1.Controls.Add(this.groupBox3, 2, 0);
            this.tableLayoutPanel1.Dock = System.Windows.Forms.DockStyle.Fill;
            this.tableLayoutPanel1.Location = new System.Drawing.Point(3, 3);
            this.tableLayoutPanel1.Name = "tableLayoutPanel1";
            this.tableLayoutPanel1.RowCount = 1;
            this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 50F));
            this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 50F));
            this.tableLayoutPanel1.Size = new System.Drawing.Size(1007, 138);
            this.tableLayoutPanel1.TabIndex = 10;
            //
            // groupBox1
            //
            this.groupBox1.Controls.Add(this.tableLayoutPanel2);
            this.groupBox1.Dock = System.Windows.Forms.DockStyle.Fill;
            this.groupBox1.Location = new System.Drawing.Point(3, 3);
            this.groupBox1.Name = "groupBox1";
```

```
            this.groupBox1.Size = new System.Drawing.Size(329, 132);
            this.groupBox1.TabIndex = 10;
            this.groupBox1.TabStop = false;
            //
            // tableLayoutPanel2
            //
            this.tableLayoutPanel2.ColumnCount = 1;
            this.tableLayoutPanel2.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 100F));
            this.tableLayoutPanel2.Controls.Add(this.rtb_sieved, 0, 1);
            this.tableLayoutPanel2.Controls.Add(this.label1, 0, 0);
            this.tableLayoutPanel2.Dock = System.Windows.Forms.DockStyle.Fill;
            this.tableLayoutPanel2.Location = new System.Drawing.Point(3, 16);
            this.tableLayoutPanel2.Name = "tableLayoutPanel2";
            this.tableLayoutPanel2.RowCount = 2;
            this.tableLayoutPanel2.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute, 15F));
            this.tableLayoutPanel2.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 100F));
            this.tableLayoutPanel2.Size = new System.Drawing.Size(323, 113);
            this.tableLayoutPanel2.TabIndex = 0;
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(3, 0);
            this.label1.Name = "label1";
```

```
            this.label1.Size = new System.Drawing.Size(30, 13);
            this.label1.TabIndex = 1;
            this.label1.Text = "Сіяні";
            //
            // groupBox2
            //
            this.groupBox2.Controls.Add(this.tableLayoutPanel3);
            this.groupBox2.Dock = System.Windows.Forms.DockStyle.Fill;
            this.groupBox2.Location = new System.Drawing.Point(338, 3);
            this.groupBox2.Name = "groupBox2";
            this.groupBox2.Size = new System.Drawing.Size(329, 132);
            this.groupBox2.TabIndex = 11;
            this.groupBox2.TabStop = false;
            //
            // tableLayoutPanel3
            //
            this.tableLayoutPanel3.ColumnCount = 1;
            this.tableLayoutPanel3.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 100F));
            this.tableLayoutPanel3.Controls.Add(this.label2, 0, 0);
            this.tableLayoutPanel3.Controls.Add(this.rtb_not_sieved, 0, 1);
            this.tableLayoutPanel3.Dock = System.Windows.Forms.DockStyle.Fill;
            this.tableLayoutPanel3.Location = new System.Drawing.Point(3, 16);
            this.tableLayoutPanel3.Name = "tableLayoutPanel3";
            this.tableLayoutPanel3.RowCount = 2;
            this.tableLayoutPanel3.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute, 15F));
```

```
            this.tableLayoutPanel3.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 100F));
            this.tableLayoutPanel3.Size = new System.Drawing.Size(323, 113);
            this.tableLayoutPanel3.TabIndex = 0;
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(3, 0);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(43, 13);
            this.label2.TabIndex = 10;
            this.label2.Text = "Несіяні";
            //
            // groupBox3
            //
            this.groupBox3.Controls.Add(this.tableLayoutPanel5);
            this.groupBox3.Dock = System.Windows.Forms.DockStyle.Fill;
            this.groupBox3.Location = new System.Drawing.Point(673, 3);
            this.groupBox3.Name = "groupBox3";
            this.groupBox3.Size = new System.Drawing.Size(331, 132);
            this.groupBox3.TabIndex = 12;
            this.groupBox3.TabStop = false;
            //
            // tableLayoutPanel5
            //
            this.tableLayoutPanel5.ColumnCount = 2;
```

```
            this.tableLayoutPanel5.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Absolute, 85F));
            this.tableLayoutPanel5.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 100F));
            this.tableLayoutPanel5.Controls.Add(this.label3, 0, 0);
            this.tableLayoutPanel5.Controls.Add(this.mtb_groups_count, 1, 0);
            this.tableLayoutPanel5.Controls.Add(this.button1, 0, 1);
            this.tableLayoutPanel5.Controls.Add(this.b_init, 1, 1);
            this.tableLayoutPanel5.Dock = System.Windows.Forms.DockStyle.Fill;
            this.tableLayoutPanel5.Location = new System.Drawing.Point(3, 16);
            this.tableLayoutPanel5.Name = "tableLayoutPanel5";
            this.tableLayoutPanel5.RowCount = 2;
            this.tableLayoutPanel5.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute, 25F));
            this.tableLayoutPanel5.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 100F));
            this.tableLayoutPanel5.Size = new System.Drawing.Size(325, 113);
            this.tableLayoutPanel5.TabIndex = 0;
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(3, 0);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(78, 13);
            this.label3.TabIndex = 0;
            this.label3.Text = "Кількість груп";
            //
```

```
            // mtb_groups_count
            //
            this.mtb_groups_count.BackColor = System.Drawing.SystemColors.ControlLight;
            this.mtb_groups_count.Location = new System.Drawing.Point(88, 3);
            this.mtb_groups_count.Mask = "0";
            this.mtb_groups_count.Name = "mtb_groups_count";
            this.mtb_groups_count.Size = new System.Drawing.Size(100, 20);
            this.mtb_groups_count.TabIndex = 1;
            this.mtb_groups_count.Text = "1";
            //
            // button1
            //
            this.button1.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Right)));
            this.button1.Location = new System.Drawing.Point(7, 87);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(75, 23);
            this.button1.TabIndex = 9;
            this.button1.Text = "Очистити";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.Clear_Click);
            //
            // tableLayoutPanel4
            //
            this.tableLayoutPanel4.ColumnCount = 1;
```

```
            this.tableLayoutPanel4.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 50F));
            this.tableLayoutPanel4.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 50F));
            this.tableLayoutPanel4.Controls.Add(this.tableLayoutPanel1, 0, 0);
            this.tableLayoutPanel4.Controls.Add(this.dgv_groups, 0, 1);
            this.tableLayoutPanel4.Controls.Add(this.tableLayoutPanel6, 0, 2);
            this.tableLayoutPanel4.Dock = System.Windows.Forms.DockStyle.Fill;
            this.tableLayoutPanel4.Location = new System.Drawing.Point(0, 0);
            this.tableLayoutPanel4.Name = "tableLayoutPanel4";
            this.tableLayoutPanel4.RowCount = 3;
            this.tableLayoutPanel4.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 25F));
            this.tableLayoutPanel4.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 25F));
            this.tableLayoutPanel4.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 50F));
            this.tableLayoutPanel4.Size = new System.Drawing.Size(1013, 577);
            this.tableLayoutPanel4.TabIndex = 11;
            //
            // dgv_groups
            //
            this.dgv_groups.AllowUserToAddRows = false;
            this.dgv_groups.AllowUserToDeleteRows = false;
            this.dgv_groups.AllowUserToOrderColumns = true;
            this.dgv_groups.AllowUserToResizeColumns = false;
            this.dgv_groups.AllowUserToResizeRows = false;
            this.dgv_groups.BackgroundColor = System.Drawing.SystemColors.ControlLight;
```

```
            this.dgv_groups.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
            this.dgv_groups.ColumnHeadersVisible = false;
            this.dgv_groups.Dock = System.Windows.Forms.DockStyle.Fill;
            this.dgv_groups.Location = new System.Drawing.Point(3, 147);
            this.dgv_groups.Name = "dgv_groups";
            this.dgv_groups.RowHeadersVisible = false;
            this.dgv_groups.Size = new System.Drawing.Size(1007, 138);
            this.dgv_groups.TabIndex = 11;
            //
            // tableLayoutPanel6
            //
            this.tableLayoutPanel6.AutoScroll = true;
            this.tableLayoutPanel6.AutoSizeMode = System.Windows.Forms.AutoSizeMode.GrowAndShrink;
            this.tableLayoutPanel6.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 0F));
            this.tableLayoutPanel6.Dock = System.Windows.Forms.DockStyle.Fill;
            this.tableLayoutPanel6.GrowStyle = System.Windows.Forms.TableLayoutPanelGrowStyle.AddColumns;
            this.tableLayoutPanel6.Location = new System.Drawing.Point(3, 291);
            this.tableLayoutPanel6.Name = "tableLayoutPanel6";
            this.tableLayoutPanel6.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 0F));
            this.tableLayoutPanel6.Size = new System.Drawing.Size(1007, 283);
            this.tableLayoutPanel6.TabIndex = 12;
            //
            // Form1
            //
```

```
        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.BackColor = System.Drawing.SystemColors.ControlDarkDark;
        this.ClientSize = new System.Drawing.Size(1013, 577);
        this.Controls.Add(this.tableLayoutPanel4);
        this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
        this.MinimumSize = new System.Drawing.Size(648, 397);
        this.Name = "Form1";
        this.tableLayoutPanel1.ResumeLayout(false);
        this.groupBox1.ResumeLayout(false);
        this.tableLayoutPanel2.ResumeLayout(false);
        this.tableLayoutPanel2.PerformLayout();
        this.groupBox2.ResumeLayout(false);
        this.tableLayoutPanel3.ResumeLayout(false);
        this.tableLayoutPanel3.PerformLayout();
        this.groupBox3.ResumeLayout(false);
        this.tableLayoutPanel5.ResumeLayout(false);
        this.tableLayoutPanel5.PerformLayout();
        this.tableLayoutPanel4.ResumeLayout(false);
        ((System.ComponentModel.ISupportInitialize)(this.dgv_groups)).EndInit();
        this.ResumeLayout(false);      }
    #endregion
    private System.Windows.Forms.RichTextBox rtb_sieved;
    private System.Windows.Forms.Button b_init;
    private System.Windows.Forms.RichTextBox rtb_not_sieved;
    private System.Windows.Forms.TableLayoutPanel tableLayoutPanel1;
```

```
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.TableLayoutPanel tableLayoutPanel2;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.TableLayoutPanel tableLayoutPanel3;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.TableLayoutPanel tableLayoutPanel4;
        private System.Windows.Forms.GroupBox groupBox3;
        private System.Windows.Forms.TableLayoutPanel tableLayoutPanel5;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.MaskedTextBox mtb_groups_count;
        private System.Windows.Forms.DataGridView dgv_groups;
        private System.Windows.Forms.TableLayoutPanel tableLayoutPanel6;
        private System.Windows.Forms.Button button1;    }}
```

Додаток Г

Клас Form1(Form1.cs)

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Windows.Forms;
```

```
namespace WindowsFormsApplication1
{   public partial class Form1 : Form
  {      List<CheckedListBox> check_boxes = new List<CheckedListBox>();
    //List<Button> buttonss = new List<Button>();
    int offset = 0;
    public Form1()
    {          InitializeComponent();       }
    private const string EMPTY = "EMPTY_EMPTY";
    /// <summary>
    /// Обработка нажатия кнопки "b_init"
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void b_init_Click(object sender, EventArgs e)
    {          offset = 0;
      string[] sieved = getPlayers(rtb_sieved);
      string[] not_sieved = getPlayers(rtb_not_sieved);
     not_sieved = RandomStringArrayTool.RandomizeStrings(not_sieved);
     sieved = RandomStringArrayTool.RandomizeStrings(sieved);
      int k= sieved.Length + not_sieved.Length;
      if (k % 2 != 0) k++;
      string[] both = new string[k]; //обьединение массивов
      Array.Copy(sieved, both, sieved.Length);
      Array.Copy(not_sieved, 0, both, sieved.Length, not_sieved.Length);
      both = RandomStringArrayTool.RandomizeStrings(both);
      int groups_count = Convert.ToInt32(mtb_groups_count.Text);
```

```
tableLayoutPanel6.ColumnCount = both.Length / 2 + 1;
if (groups_count == 0) return;
if (sieved.Length == 0 && not_sieved.Length == 0) return;
if ((sieved.Length + not_sieved.Length) % groups_count != 0)
{           if (MessageBox.Show("Количество груп не кратно количеству участников.\nВ группах может быть разное количество участников?", "", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
    {               int delta = groups_count - not_sieved.Length % groups_count;
        string[] tmp = new string[not_sieved.Length + delta];
        for (int i = 0; i < not_sieved.Length; i++)
        {                   tmp[i] = not_sieved[i];                 }
        for (int i = not_sieved.Length; i < tmp.Length; i++)
        {                   tmp[i] = EMPTY;
        }               not_sieved = tmp;
    }           else            {
        return;             }       }
if (sieved.Length != 0 && sieved.Length % groups_count != 0)        {
    MessageBox.Show("Количество сеяных не кратно количеству груп", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    return;         }
dgv_groups.Columns.Clear();
for (int i = 0; i < groups_count; i++) { dgv_groups.Columns.Add("group_" + (i + 1).ToString(), "Группа " + (i + 1).ToString()); }
dgv_groups.Rows.Add((sieved.Length + not_sieved.Length)/groups_count);
        output(dgv_groups, sieved, 0, groups_count);
output(dgv_groups, not_sieved, sieved.Length / groups_count, groups_count);
        tableLayoutPanel6.Controls.Clear();
output2(tableLayoutPanel6, both, offset, both.Length);
```

```
        AddButton(tableLayoutPanel6, both, offset, both.Length);
      for (int i = 0; i < dgv_groups.Rows.Count; i++)          {
          for (int j = 0; j < dgv_groups.Columns.Count; j++)
          {                   if (dgv_groups.Rows[i].Cells[j].Value.ToString() == EMPTY)
dgv_groups.Rows[i].Cells[j].Value = "";
          }          }       }
      /// <summary>
      /// Вывод массива участников
      /// </summary>
      /// <param name="dgv">Куда выводить</param>
      /// <param name="ar">Массив участников</param>
      /// <param name="offset">С какой строчки вписывать</param>
      /// <param name="gr_c">Количество столбцов</param>
      private void output(DataGridView dgv, string[] ar, int offset, int gr_c)
      {           int g = ar.Length / gr_c;
        for (int i = 0; i < gr_c; i++)
        {                 dgv_groups.Columns[i].Width = dgv_groups.Width / gr_c - 1;
          for (int j = 0; j < g; j++)
          {                   dgv_groups.Rows[j + offset].Cells[i].Value += ar[i * g + j];
          }          }       }
      /// <summary>
      /// Создание сетки
      /// </summary>
      /// <param name="tableLayout">Куда выводить</param>
      /// <param name="ar">Массив участников</param>
      /// <param name="offset">С какого столбца вписывать</param>
      /// <param name="mc_c">Количество строк</param>
```

```
        private void output2(TableLayoutPanel tableLayout, string[] ar, int offset, int mc_c)       {
          /* Column Index , Row index */
           int number = 0;
          for (int j = 0; j < (ar.Length/2); j++) //вывод боксов
            {// добавляем чеклист
              CheckedListBox newBox = new CheckedListBox();
              check_boxes.Add(newBox);
              newBox.Name = "newBox" + j;
              newBox.CheckOnClick = true;
              tableLayout.Controls.Add(newBox, offset, j);
              // заполнение боксов
              if (ar[number] != null)  newBox.Items.Add(ar[number]); number++;
              if (ar[number] != null)  newBox.Items.Add(ar[number]); number++;          }      }
        /// <summary>
        /// Создание сетки
        /// </summary>
        /// <param name="tableLayout">Куда выводить</param>
        /// <param name="ar">Массив участников</param>
        /// <param name="offset">номер стобца</param>
        /// <param name="q">номер кнопки?</param>
        private void AddButton(TableLayoutPanel tableLayout, string[] ar, int offset, int q)       {          Button
button = new Button();
            button.UseVisualStyleBackColor = true;
            button.Text = "Далі";
            button.Name = "btn" + offset;
            tableLayout.Controls.Add(button, offset, ((ar.Length) / 2));
```

```
button.Click += new System.EventHandler(this.NextButton_Click); } private
 void NextButton_Click(object sender, EventArgs e) {
    int q = 0; //считает позицию вставки
int k = 0;//кол-во выбраных елементов
//считаем количество выбраных елементов
for (int i = 0; i < check_boxes.Count; i++)
{ if (check_boxes[i].CheckedItems.Count != 0) {
   k += check_boxes[i].CheckedItems.Count; } }
if (k > 0) offset++; else
{ MessageBox.Show("Данні відсутні"); return; };
if (k % 2 !=0) k++;
string[] s = new string[k];
for (int i = 0; i < check_boxes.Count; i++)
{ if (check_boxes[i].CheckedItems.Count != 0) {
    string[] x = new string[check_boxes[i].CheckedItems.Count];
    for (int j = 0; j < check_boxes[i].CheckedItems.Count; j++)
    { x[j] = check_boxes[i].CheckedItems[j].ToString(); }
Array.Copy(x, 0, s, q, x.Length);
    q += x.Length; } }
s = RandomStringArrayTool.RandomizeStrings(s);
if (s.Length == 2 && s[0] == null || s[1] == null)
{ MessageBox.Show("Команда '" + s[0] + s[1] + "' перемогла!"); return; }
check_boxes.Clear();
output2(tableLayoutPanel6, s, offset, s.Length);
AddButton(tableLayoutPanel6, s, offset, s.Length); }
/// <summary>
/// Получение списка участников из текстового поля
```

```
/// </summary>
/// <param name="rtb">Текстовое поле, с которого идет выборка</param>
/// <returns>Массив участников</returns>
private string[] getPlayers(RichTextBox rtb)
{          Regex r = new Regex("(\n){1,}");
   rtb.Text = r.Replace(rtb.Text, "\n");
   if (rtb.Text.Length == 0)
   {              return new string[0];          }
   string cn = rtb.Text.Replace("\n", "|").Replace("\r", "|");
   cn = cn.Replace("||", "|");
   if (cn[cn.Length - 1] == '|')
      cn = cn.Substring(0, cn.Length - 1);
   string[] tmp = explode("|", cn);
   return tmp;       }
/// <summary>
/// Вывод случайно сгенерированных групп
/// </summary>
/// <param name="indx">Индекс группы</param>
/// <param name="rtb">Куда выводить</param>
/// <param name="strings">Массив участников</param>
/// <param name="c">Количество на одну группу</param>
/*   private void input_rtb(int indx, RichTextBox rtb, string[] strings, int c)
{          for (int i = 0; i < c; i++)
   {              rtb.Text += strings[indx * c + i] + "\n";
   }       }*/
private void Clear_Click(object sender, EventArgs e)
```

```
{           tableLayoutPanel6.Controls.Clear();
   offset = 0;       }
/// <summary>
/// Разбить строку на подстроку через указанный разделитель
/// </summary>
/// <param name="separator">Разделитель</param>
/// <param name="source">Разбиваемая строка</param>
/// <returns>Массив подстрок</returns>
private string[] explode(string separator, string source)
{           return source.Split(new string[] { separator }, StringSplitOptions.None);       }    }}
```