О. ШИНКАРУК, Н. БИШЕВЕЦЬ,
К. СЕРГІЄНКО, О. ЯКОВЕНКО, В. УСИЧЕНКО

# ОСНОВИ ПРОГРАМУВАННЯ, СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОБУДОВА КОМП'ЮТЕРНИХ СИСТЕМ

НАВЧАЛЬНИЙ ПОСІБНИК

Київ
Національний університет фізичного
виховання і спорту України
«Олімпійська література», 2024

**УДК 519.68+004.4+004.03+032.2**

*Затверджено до друку Вченою радою Національного університету фізичного виховання і спорту України, протокол № 9 від 29.02.2024 р.*

**Рецензенти:**

*Кисельов В. Б.* – доктор технічних наук, професор, директор Навчально-наукового інституту муніципального управління та міського господарства ТНУ ім. В. І. Вернадського;

*Омецинська Н. В.* – кандидат технічних наук, доцент, завідувач кафедри загальноінженерних дисциплін та теплоенергетики Навчально-наукового інституту муніципального управління та міського господарства ТНУ ім. В. І. Вернадського;

*Шевчук О. М.* – кандидат наук з фізичного виховання і спорту, доцент, доцент кафедри кінезіології та фізкультурно-спортивної реабілітації Національного університету фізичного виховання і спорту України

**Шинкарук О., Бишевець Н., Сергієнко К., Яковенко О., Усиченко В.**

Основи програмування, створення програмного забезпечення та побудова комп'ютерних систем: навчальний посібник. – К.: Національний університет фізичного виховання і спорту України, вид-во «Олімп. л-ра, 2024. – 140 с.

ISBN 978-617-7492-28-2

В основу навчального посібника покладено багаторічний досвід викладання дисциплін «Теорія ймовірностей та математична статистика», «Математичне програмування», «Дослідження операцій», «Інноваційні та інформаційні технології у фізичній культурі і спорті».

Зміст навчального посібника узгоджено з робочою програмою навчальної дисципліни «Основи програмування, створення програмного забезпечення та побудова комп'ютерних систем». Цикл лекцій та практичних завдань спрямований на формування визначених освітньо-професійною програмою загальних і фахових компетентностей, формування алгоритмічного та логічного мислення, здатності виявляти та ефективно розв'язувати складні спеціалізовані задачі та практичні проблеми інноваційного й наукового характеру в кіберспорті (esports), ознайомлення з інструментальними програмними та технічними рішеннями, які використовуються в кіберіндустрії. Ознайомлення зі змістом навчального посібника дозволить сформувати систему знань щодо мов програмування й їх еволюції, створити підгрунтя для подальшого оволодіння базовими поняттями та прийомами програмування, оволодіти сучасним рівнем інформаційної і комп'ютерної культури, теоретичними основами й практичними навичками розробки алгоритмічного та програмного забезпечення, підготовки до усвідомленого використання програмного забезпечення для розв'язання конкретних задач у сфері кіберспорту. Як універсальний інструмент для обчислень у посібнику використовується популярний табличний процесор MS Excel.

Для здобувачів вищої освіти зі спеціалізації кіберспорт другого (магістерського) рівня освіти, науково-педагогічних працівників закладів вищої освіти, тренерів, спортсменів, спеціалістів з кіберспорту.

**УДК 519.68+004.4+004.03+032.2**

© Національний університет фізичного виховання і спорту України, видавництво «Олімпійська література», 2024

© О. Шинкарук, Н. Бишевець, К. Сергієнко, О. Яковенко, В. Усиченко, 2024

ISBN 978-617-7492-28-2

# ЗМІСТ

# ВСТУП

У сучасному суспільстві рівень оволодіння фахівцем навичками застосування інформаційних технологій (ІТ) для вирішення професійно орієнтованих завдань разом з іншими значущими компетентностями визначає його конкурентоспроможність на ринку праці. Серед роботодавців стає дедалі більш затребуваним досвід вирішення практичних завдань засобами ІТ, причому це стосується всіх фахівців незалежно від сфери їхньої професійної діяльності. Підготовка майбутніх фахівців сфери фізичного виховання і спорту не є винятком у цих процесах глобальної інформатизації та потребує системного підходу до трансформації змісту освіти.

Особливо актуальним питанням, спрямованим на удосконалення змісту освіти майбутніх фахівців сфери кіберспорту, є застосування накопиченого досвіду в процесі їхньої фахової підготовки.

На етапі започаткування й становлення кіберспортивної науки важливою складовою підготовки фахівців у сфері кіберспорту є розширення теоретичної бази та формування алгоритмічного мислення майбутніх фахівців, яке тісно пов'язане з вирішенням питань програмування. Відтак у навчальному посібнику чільне місце займає розділ, присвячений розробці алгоритмів. Оволодіння викладеним матеріалом дозволить свідомо обрати напрям удосконалення в питаннях розробки комп'ютерних програм.

Роль статистичного аналізу результатів досліджень у різних галузях знань посилюється. Накопичення даних у сфері кіберспорту вимагає їх систематизації й узагальнення. Застосування методів статистичного аналізу дозволить отримати науково обґрунтовану інформацію про основні тенденції розвитку кіберспорту, визначити вплив факторів на ефективність кіберспортсменів, узагальнити дані про фізичні, інтелектуальні, психоемоційні особливості провідних вітчизняних та світових кіберспортсменів.

Застосування сучасних інформаційних технологій для статистичної обробки даних у сфері кіберспорту є найбільш перспективним напрямом оптимізації процесу розвитку кіберспортивної науки. Оволодіння методами статистичної обробки інформації засобами ІТ відкриває можливості майбутнім фахівцям сфери кіберспорту здійснювати якісний аналіз результатів досліджень без поглибленої математичної підготовки. При цьому позитивний досвід вирішення професійно орієнтованих завдань сприяє підвищенню мотивації у майбутніх фахівців сфери кіберспорту використовувати засвоєні методи у дослідницькій практиці.

Крім того, існують перспективи застосування методів математичного програмування у сфері кіберспорту.

Як ми пересвідчилися, застосування математичних методів та підходів дозволяє оптимізувати подальший розвиток явищ і процесів у кіберспорті. Їх доцільно використовувати у ході вирішення широкого кола завдань у кіберспорті, зокрема під час:

- планування та управління підготовкою кіберспортсменів;
- вироблення та прийняття управлінських рішень тренером або управлінцями;
- оптимального вибору каналу поширення реклами кіберспортивних заходів тощо.

Надбудова Розв'язувач, реалізована в Excel, автоматизує процес прийняття рішення.

Матеріал, викладений у навчальному посібнику, – практико-орієнтований. Його рекомендується вивчати в процесі виконання запропонованих завдань. Сподіваємося, що опанування змісту дисципліни стане поштовхом для подальшого розвитку та професійного вдосконалення майбутніх фахівців у сфері кіберспорту.

ТЕМА 1

# МОВИ ПРОГРАМУВАННЯ. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПРИКЛАДНИХ ЗАДАЧ У ФІЗИЧНІЙ КУЛЬТУРІ І СПОРТІ. СУЧАСНІ КОМП'ЮТЕРНІ СИСТЕМИ В КІБЕРСПОРТІ

## ЛЕКЦІЯ 1. МОВИ ПРОГРАМУВАННЯ

***1. Мови програмування і синтаксис.***
***2. Основні характеристики об'єктів даних і типів даних.***
***3. Поняття парадигми програмування.***
***4. Мови програмування низького і високого рівнів.***
***5. Історія розвитку мов програмування.***
***6. П'ять поколінь мов програмування.***
***7. Мови програмування комп'ютерних ігор.***

### МОВИ ПРОГРАМУВАННЯ І СИНТАКСИС

Мова програмування допомагає людині записати зрозумілу для комп'ютера послідовність операцій.

**Мова програмування** – це мова, якою описуються алгоритми та структури даних, або, іншими словами, засіб, який забезпечує вираження алгоритмів.

Понад дві з половиною тисячі мов програмування було створено з початку розробки перших програмованих машин. У процесі професійної діяльності кваліфіковані спеціалісти зазвичай використовують кілька мов програмування. Майже всі вони мають головне завдання – це перетворення зрозумілих людині команд у машинний код, що потребує чітко визначеного синтаксису.

**Синтаксис** мови програмування – набір правил, які визначають, як написати програмний код, щоб комп'ютер міг його зрозуміти та виконати. Кожна мова програмування має свій унікальний синтаксис, який складається з команд, операторів, функцій і структур даних.

#### Основні елементи синтаксису мови програмування

• *Ключові слова* – це слова, які мають спеціальне значення в мові програмування. Їх використовують для позначення різних елементів мови, таких як типи даних, оператори, конструкції тощо. Наприклад, у мові **C** ключовими словами є int, float, if, else тощо.

• *Ідентифікатори* – це імена, які використовують для позначення змінних, функцій, класів тощо. Вони повинні починатися з букви або знака підкреслення і можуть містити букви, цифри та знаки підкреслення. Наприклад, у мові **C** ідентифікаторами можуть бути x, y, z, my_function, my_class тощо.

• *Оператори* – це символи, які використовують для виконання операцій над даними. Наприклад, у мові **C** оператори додавання, віднімання, множення та ділення позначають символами +, –, * та / відповідно.

• *Вирази* – це комбінації операторів, ідентифікаторів та чисел, які обчислюються комп'ютером. Наприклад, вираз x + 1 обчислює суму змінної x та числа 1.

• *Команди* – це послідовності операторів, які інструктують комп'ютер виконати певні дії. Наприклад, команда print(x) інструктує комп'ютер вивести значення змінної x на екран.

Синтаксис мови програмування є важливою її частиною. Він визначає, як програмісти можуть спілкуватися з комп'ютером, щоб інструктувати його виконувати певні завдання. Якщо синтаксис коду не відповідає правилам мови програмування, комп'ютер не зможе його зрозуміти та виконати. Наведемо кілька прикладів синтаксису у різних мовах програмування (приклад оголошення змінної, де ключове слово x оголошує змінну; знак рівності (=) присвоює змінній значення; число 10 є значенням змінної):

| **Python** | **Java** | **C++** |
|---|---|---|
| x = 10 | int x = 10; | int x = 10 |

Приклад програмного коду у Visual Basic представлено на рисунку (рис. 1.1).

Ці приклади демонструють, як синтаксис мови програмування використовується для інструктування комп'ютера виконувати певні завдання.

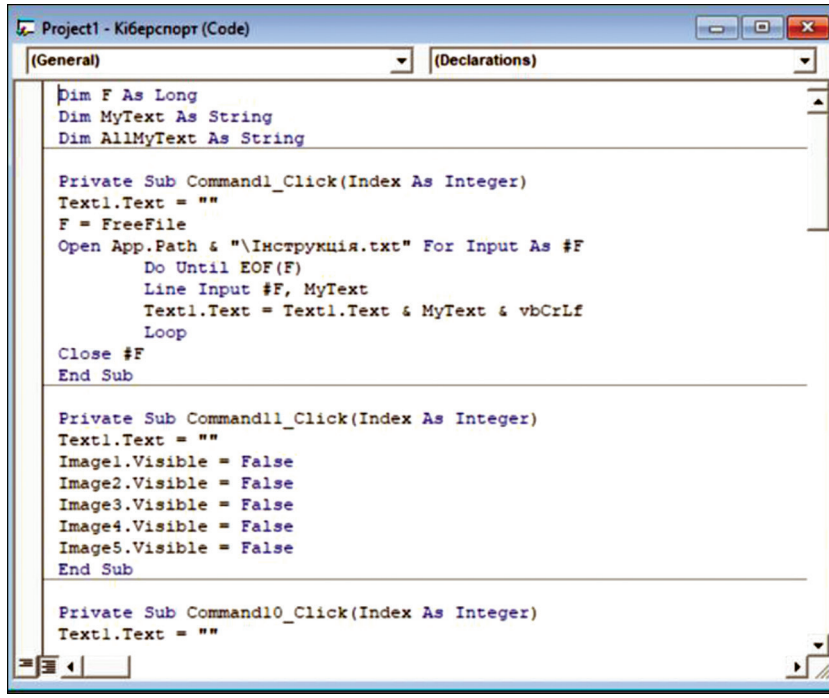



**Рисунок 1.1** – Приклад частини програмного коду у Visual Basic

## Основні характеристики об'єктів даних

У фізичній пам'яті комп'ютера дані зберігаються у вигляді послідовності бітів, які групуються в байти або слова. Це проста структура, яка дозволяє швидко і ефективно отримувати доступ до даних.

Однак у віртуальному комп'ютері, який не має власного апаратного забезпечення, дані зберігаються у більш складних формах, таких як стеки, масиви, числа, символьні рядки тощо. Ці форми зберігання даних дозволяють віртуальному комп'ютеру реалізовувати більш складні функції, ніж фізичний комп'ютер. У віртуальному комп'ютері дані можуть бути об'єднані в групи, які називають об'єктами даних. Об'єкт даних може містити один або кілька однотипних елементів даних.

**Об'єкти даних** – це дані, які зберігаються разом. Об'єкт даних може бути будь-якого розміру – від одного елемента до мільйонів елементів. Об'єкти даних дозволяють зберігати й обробляти дані. Наприклад, їх можна використовувати для зберігання чисел, тексту, логічних значень, дат і часу тощо, також для зберігання складніших структур даних, таких як масиви, словники та структури.

**Тип даних** – це визначення того, які дані можуть зберігатися в об'єкті даних. Він також визначає, які операції можна виконувати з об'єктом даних. У кожній мові програмування є вбудовані типи даних, які призначені для зберігання різних типів даних. Наприклад, в **C++** є вбудовані типи даних для зберігання цілих чисел, дійсних чисел, символів та рядків. Типи даних – це те, що визначає, які дані можуть зберігатися в об'єкті даних. Наприклад, тип даних «ціле число» може зберігати лише цілі числа, а тип даних «символьний рядок» – лише рядки символів.

Типи даних дозволяють комп'ютеру розуміти, які дані зберігаються в об'єкті даних і дозволяють виконувати правильні операції з об'єктами даних. Наприклад, якщо комп'ютер знає, що об'єкт даних є типом «ціле число», то він може виконувати з ним операції додавання, віднімання, множення та ділення. Однак, якщо комп'ютер не знає, що об'єкт даних є типом «ціле число», то він може помилитися, наприклад, спробувати додати об'єкт даних до рядка тексту.

Наведемо приклади використання об'єктів даних і типів даних у мовах програмування.

- У програмі для обчислення квадратного кореня можна використовувати об'єкт даних типу «дійсне число» для зберігання значення числа, з якого потрібно обчислити квадратний корінь.
- У програмі для сортування даних можна використовувати об'єкти даних типу «рядок тексту» для зберігання даних, які потрібно відсортувати.
- У програмі для управління базою даних можна використовувати об'єкти даних типу «структура» для зберігання інформації про запис у базі даних.

Об'єкти даних і типи даних – це фундаментальні поняття в мовах програмування, і тому важливо розуміти, що це таке і як їх використовувати в програмуванні.

Характеристики об'єктів даних і типів даних у мовах програмування представлено у таблиці (табл. 1.1)

**Таблиця 1.1** – Основні характеристики об'єктів даних і типів даних

| Характеристика | Об'єкт даних | Тип даних |
|---|---|---|
| Склад об'єкта даних визначає, з яких елементів він складається. Наприклад, число 123 є об'єктом даних типу «ціле число», що складається з трьох цифр | Може складатися з одного або кількох однотипних елементів даних | Визначає, які дані можуть зберігатися в об'єкті даних |
| Розмір визначає, скільки пам'яті він займає. Наприклад, тип даних «ціле число» займає 4 байти пам'яті, а тип даних «рядок тексту» може займати від 1 байта до мільйонів байт пам'яті | Може бути будь-якого розміру, від одного елемента до мільйонів елементів | Може займати від 1 байта до мільйонів байтів пам'яті |
| Види можна розділити на прості і складні. Прості об'єкти даних складаються з одного елемента, а складні можуть складатися з кількох простих об'єктів даних | Прості, складні | Вбудовані, користувальницькі |
| Призначення об'єктів даних і типів даних визначається тим, як вони використовуються в програмі. Об'єкти даних використовуються для зберігання і обробки даних, а типи даних – для визначення того, які операції можна виконувати з об'єктом даних | Зберігання і обробка даних | Визначення того, які операції можна виконувати з об'єктом даних |

## ПОНЯТТЯ ПАРАДИГМИ ПРОГРАМУВАННЯ

**Парадигма програмування** – це система ідей і понять, які визначають стиль написання комп'ютерних програм, а також спосіб мислення програміста. Парадигма програмування – спосіб концептуалізації, що визначає організацію обчислень і структурування роботи, яку виконує комп'ютер.

Поняття парадигми програмування було введено в 1950-х роках математиком і програмістом Джоном Маккарті, який вважав, що мови програмування можна класифікувати за тим, наскільки вони віддалені від машинного коду.

Головним у парадигмі програмування є те, як програміст мислить про проблему, яку вирішує. Кожна парадигма має свої власні особливості, що впливають на те, як програміст організовує код і вирішує проблеми.

### Основні парадигми програмування

Мови програмування можна класифікувати за різними ознаками. За парадигмою програмування їх поділяють на:

**Об'єктно-орієнтоване програмування** (ООП) – це парадигма програмування, яка базується на понятті об'єкта. Об'єкт – це сукупність даних і поведінки, яка може взаємодіяти з іншими об'єктами.

**Функціональне програмування** – це парадигма програмування, що базується на понятті функції. Функція – це математична функція, яка приймає на вхід деякі значення і повертає деяке значення.

**Процедурне програмування** – це парадигма програмування, що базується на понятті процедури. Процедура – це послідовність операторів, які виконуються як єдине ціле.

**Логічні мови програмування** – це парадигма програмування, яка базується на понятті логіки. Логічні мови програмування використовують для вирішення завдання логічного виведення.

У таблиці 1.2 візуалізовано узагальнену інформацію про основні парадигми програмування, їхні визначальні концепції, характеристики та приклади мов програмування.

***Об'єктно-орієнтоване програмування*** (ООП) є однією з найпопулярніших парадигм програмування. Добре підходить для вирішення завдань, у яких потрібно моделювати реальний світ. Наприклад, ООП можна використовувати для створення програм, які моделюють роботу підприємства, ігор або вебсайтів та взаємодіяти з іншими об'єктами.

Основні концепції ООП:

- *Інкапсулювання* – об'єднання даних і поведінки в єдиний об'єкт.
- *Наслідування* – можливість створення нового об'єкта, який успадковує властивості і поведінку від іншого об'єкта.
- *Абстракція* – можливість розглядати об'єкти як сукупність їх властивостей і поведінки, не вдаючись до деталей їх реалізації.
- *Поліморфізм* – можливість використовувати один і той самий код для роботи з різними об'єктами.

Приклади мов програмування, які підтримують ООП:

- Java;
- C++;
- Python;
- JavaScript.

Розглянемо приклад створення програми, яка моделює роботу підприємства. У цій програмі можна створити об'єкти для таких елементів, як працівники, відділи, продукти та клієнти. Об'єкти працівників будуть мати такі властивості, як ім'я, посада та зарплата, об'єкти відділів – назва, номер і керівник, об'єкти продуктів – ціна та кількість, об'єкти клієнтів – ім'я, адреса та номер телефону.

Ці об'єкти можуть взаємодіяти один з одним. Наприклад, працівник може бути закріплений за відділом, а відділ може виробляти продукти. Продукт може бути проданий клієнту.

**Таблиця 1.2** – Основні парадигми програмування, їхні визначальні концепції, характеристики та приклади мов програмування

| Парадигма | Основна концепція | Характеристика | Приклади |
|---|---|---|---|
| Процедурне програмування | Процедура | Фокус на послідовності інструкцій | C, C++, Java, Python |
| Об'єктно-орієнтоване програмування | Об'єкт | Фокус на взаємодії об'єктів | Java, C++, Python, JavaScript |
| Функціональне програмування | Функція | Фокус на обчисленні функцій | Haskell, Lisp, Clojure, Python |
| Логічні мови програмування | Логіка | Фокус на логічному обчисленні | Prolog, Datalog, Clingo |

ООП дозволяє створювати програми, які є більш гнучкими, повторюваними та читабельними.

*Переваги ООП:*

• збільшується читабельність і зрозумілість коду. ООП дозволяє групувати дані і поведінку в єдиний об'єкт, що робить код більш зрозумілим і читабельним;

• збільшується повторюваність коду. ООП дозволяє використовувати наслідування для створення нових об'єктів, які успадковують властивості і поведінку від інших об'єктів. Це дозволяє зменшити обсяг коду, який потрібно написати.

• збільшується гнучкість коду. ООП дозволяє використовувати поліморфізм для роботи з різними об'єктами за допомогою одного і того самого коду. Це дозволяє зробити код більш гнучким і адаптивним.

*Недоліки ООП:*

• ООП може бути складним для розуміння. Воно є потужною парадигмою, але може бути складним для розуміння початківцям;

• ООП може призвести до надмірного використання абстракції, дозволяє створювати абстрактні об'єкти, які представляють різні речі, однак надмірне використання абстракції може зробити код менш зрозумілим і читабельним.

*Висновок*:

ООП – це потужна парадигма програмування, що має ряд переваг. ООП використовують у широкому спектрі програмних проєктів, включаючи вебсайти, мобільні додатки та ігри.

***Функціональне програмування.***

Основні концепції функціонального програмування:

*Імутабельність* – вимога, щоб дані не змінювалися в процесі виконання програми.

*Функції* першого класу – функції, які можуть бути передані як аргументи до інших функцій, повернуті з інших функцій і призначені для змінних.

*Рекурсія* – техніка програмування, в якій функція викликає сама себе.

*Лямбда-вирази* – короткі функції, які можуть бути визначені безпосередньо в коді.

Приклади мов програмування, які підтримують функціональне програмування:

• Haskell;
• OCaml;
• Erlang;
• Python.

Розглянемо приклад створення програми, яка сортує список чисел. У традиційному підході ми можемо використовувати цикл for або while для вибору елементів списку і порівняння їх один з одним. Цей підхід може бути ефективним для невеликих списків, але стає неефективним для великих.

У функціональному підході можна використовувати рекурсію для сортування списку. Рекурсивна функція сортування приймає на вхід список і повертає список, в якому елементи сортовані. Функція працює шляхом поділу списку на два підписи, сортування кожного підспису окремо, а потім об'єднання двох сортованих підписів в один сортований список.

Наводимо приклад рекурсивної функції сортування:

Python

```
def sort(numbers):
  if len(numbers) <= 1:
    return numbers

  middle = len(numbers) // 2
  left = sort(numbers[:middle])
  right = sort(numbers[middle:])
  return left + right
```

Ця функція працює шляхом поділу списку на два підписи з рівною кількістю елементів. Потім вона викликає саму себе для сортування кожного підспису. Нарешті, вона об'єднує два сортованих підписи в один сортований список.

Функціональне програмування дозволяє нам створювати програми, які є більш ефективними, простими в розумінні та читабельними.

*Переваги функціонального програмування:*

• ефективність. Функціональні програми часто більш ефективні, ніж програми, написані в інших парадигмах, таких як ООП. Це пов'язано з тим, що вони використовують рекурсію, яка є ефективним способом обробки даних;

• простота в розумінні. Функціональні програми часто простіші в розумінні, ніж програми, написані в інших парадигмах. Це пов'язано з тим, що вони використовують функції першого класу, які полегшують читання та розуміння коду;

• читабельність. Функціональні програми часто більш читабельні, ніж програми, написані в інших парадигмах. Це пов'язано з тим, що вони використовують імутабельність (незмінність), яка полегшує відстеження стану програми.

*Недоліки функціонального програмування:*

• складність. Функціональне програмування може бути складним для розуміння початківцям. Це пов'язано з тим, що воно використовує ряд нових понять, таких як імутабельність і функції першого класу;

• обмеження. Функціональне програмування може бути обмеженим для деяких завдань. Наприклад, його складно використовувати для створення інтерфейсів користувача.

Функціональне програмування – це потужна парадигма програмування, яка має ряд переваг. Функціональне програмування використовується в різних сферах, включаючи розробку веб-сайтів, мобільних додатків і ігор.

***Процедурне програмування.***

Основні концепції процедурного програмування:

*Декомпозиція* – процес поділу складної проблеми на менш складні підзадачі.

*Ітеративність* – техніка програмування, в якій одна і та сама послідовність операторів виконується повторно, поки не буде досягнуто певного стану.

*Рекурсія* – техніка програмування, в якій функція викликає сама себе.

Приклади мов програмування, які підтримують процедурне програмування:

- C;
- C++;
- Java;
- Python;

Розглянемо приклад створення програми, яка сортує список чисел. У процедурному підході можемо використовувати цикл for або while для вибору елементів списку і порівняння їх один з одним.

Наводимо приклад процедури сортування:

C

```
void sort(int* numbers, int size) {
  for (int i = 0; i < size; i++) {
    for (int j = i + 1; j < size; j++) {
      if (numbers[i] > numbers[j]) {
        int temp = numbers[i];
        numbers[i] = numbers[j];
        numbers[j] = temp;
      }
    }
  }
}
```

Ця процедура працює шляхом вибору елементів списку, починаючи з першого елемента. Для кожного елемента вона перебирає всі наступні елементи, порівнюючи їх з поточним елементом. Якщо поточний елемент більший наступного елемента, то їхні значення міняються місцями. Це дозволяє створювати програми, які є простими в розумінні та читабельними.

*Переваги процедурного програмування:*

- простота в розумінні. Програми, написані в процедурному стилі, часто простіші в розумінні, ніж програми, написані в інших парадигмах. Це пов'язано з тим, що воно використовує прості концепції, такі як процедури та цикли;
- читабельність. Програми, написані в процедурному стилі, часто читабельніші, ніж програми, написані в інших парадигмах. Це пов'язано з тим, що воно використовує логічну структуру, яка полегшує відстеження потоку виконання програми.

*Недоліки процедурного програмування:*

• неефективність. Програми, написані в процедурному стилі, можуть бути неефективними, порівняно з програмами, написаними в інших парадигмах, таких як ООП або функціональне програмування. Це пов'язано з тим, що процедурне програмування часто використовує цикли, які можуть бути неефективними для обробки великих даних;

• складність. Програми, написані в процедурному стилі, можуть бути складними для розуміння, якщо вони великі і складні.

Процедурне програмування – це проста і потужна парадигма програмування, яка має ряд переваг. Його використовують у різних сферах, включаючи розробку операційних систем, драйверів пристроїв і інших системних програм.

***Логічні мови програмування*** – це парадигма програмування, яка базується на понятті логіки. Логіка – це наука про правильне мислення, і логічні мови програмування використовують логічні правила для визначення того, що є правильним і неправильним у програмі.

Основні концепції логічного програмування:

*Логічні вирази* – вирази, які можуть бути оцінені як істинні або хибні.

*Логічні оператори* – оператори, які можуть бути використані для об'єднання логічних виразів.

*Запитання* – логічні вирази, які можуть бути оцінені як істинні або хибні.

*Програми* – набори правил, які визначають, що є правильним і неправильним у програмі.

Приклади мов програмування, які підтримують логічне програмування:

• Prolog – перша і найвідоміша мова логічного програмування;

• Datalog – мова логічного програмування, яка використовується для маніпулювання даними;

• Clingo – мова логічного програмування, яка використовується для розв'язання задач логічного планування.

Розглянемо приклад створення програми, яка визначає, чи є число парним. У логічному підході ми можемо використовувати такий запит:

```
Prolog
is_even(X) :– X mod 2 = 0.
```

Цей запит означає, що число X є парним, якщо воно ділиться на 2 без залишку.

Можемо використовувати цей запит для визначення, чи є число парним, наприклад, так:

```
Prolog
?– is_even(10).
```

Цей запит поверне значення true, оскільки 10 ділиться на 2 без залишку.

*Переваги логічних мов програмування:*

• простота. Логічні мови програмування часто більш прості в розумінні та читабельні, ніж програми, написані в інших парадигмах;

• ефективність. Логічні мови програмування можуть бути ефективними для обробки великих наборів даних;

• експансивність. Логічні мови програмування можна використовувати для вирішення широкого спектра задач.

*Недоліки логічних мов програмування:*

• складність. Логічні мови програмування можуть бути складними для розуміння, якщо вони великі і складні;

• негнучкість. Логічні мови програмування можуть бути негнучкими для вирішення деяких задач.

Логічні мови програмування використовують у різних сферах, включаючи розробку баз даних, штучного інтелекту та логічного планування.

Парадигми програмування виникли з розвитком комп'ютерної техніки. Перші мови програмування, такі як асемблер, були низькорівневими і використовували процедурне програмування. Цей підхід був простим і зрозумілим, але він не дозволяв ефективно моделювати реальний світ.

У 1960-х роках з'явилося об'єктно-орієнтоване програмування. ООП було розроблене для вирішення проблем, які виникали при використанні процедурного програмування. Воно дозволило об'єднувати дані і поведінку в єдині об'єкти, що зробило код більш зрозумілим і повторюваним.

У 1970-х роках з'явилося функціональне програмування, яке було розроблене для вирішення проблем, пов'язаних з паралельним програмуванням і роботою з великими обсягами даних. Функціональне програмування дозволяє уникати побічних ефектів, що робить код більш надійним і безпечним.

У 1980-х роках з'явилося логічне програмування, розроблене для розв'язання задач, пов'язаних зі штучним інтелектом. Воно дозволяє використовувати логічні правила для визначення того, що є правильним і неправильним у програмі.

## Практичні поради до вибору парадигми програмування

Під час вибору парадигми програмування слід враховувати такі фактори:

• Тип задачі, яку потрібно розв'язати. ООП добре підходить для розв'язання задач, в яких потрібно моделювати реальний світ. Функціональне програмування добре підходить для задач, пов'язаних з паралельним програмуванням і роботою з великими обсягами даних. Процедурне програмування добре підходить для розв'язання простих задач. Логічні мови програмування добре підходять для задач, пов'язаних зі штучним інтелектом.

• Розмір і складність програми. ООП може бути складним для розуміння, особливо для великих і складних програм. Функціональне програмування може бути більш ефективним для великих і складних програм, але воно може бути більш складним для розуміння, ніж ООП.

• Особливі вимоги до програми. Наприклад, якщо програма має бути ефективною для паралельного виконання, то слід використовувати ООП або функціональне програмування. Якщо програма повинна бути надійною і безпечною, то слід використовувати функціональне програмування.

Узагальнюючи, можна сказати, що ООП є універсальною парадигмою, яка підходить для розв'язання широкого спектра задач. Функціональне програму-

вання є потужною парадигмою, яка може бути ефективнішою, ніж ООП для деяких типів задач. Процедурне програмування є простою і зрозумілою парадигмою, яка підходить для розв'язання простих задач. Логічні мови програмування є спеціалізованою парадигмою, яка підходить для задач, пов'язаних зі штучним інтелектом.

За сферою застосування мови програмування поділяють на універсальні та спеціалізовані.

**Універсальні мови програмування** можна використовувати для розв'язання широкого спектра задач. Вони містять широкий набір мовних засобів, що дозволяють програмувати на різних рівнях абстракції.

Прикладами універсальних мов програмування є:

- C++;
- Java;
- Python;
- JavaScript.

**Спеціалізовані мови програмування** призначені для розв'язання конкретних задач. Вони містять обмежений набір мовних засобів, що дозволяють програмувати на конкретному рівні абстракції.

Прикладами спеціалізованих мов програмування є:

- SQL – мова для роботи з базами даних;
- HTML – мова для створення веб-сторінок;
- CSS – мова для стилізації веб-сторінок;
- JavaScript – мова для створення інтерактивних веб-сторінок;
- C – мова для програмування низького рівня;
- Matlab – мова для наукових обчислень.

Незважаючи на те що спеціалізовані мови програмування теж бувають Тьюринг-повні, тобто можуть використовуватися для розв'язання будь-яких задач, їх сфера застосування обмежена. Це пов'язано з тим, що вони містять обмежений набір мовних засобів, призначених для розв'язання конкретних задач. Наприклад, мова shell призначена для взаємодії з операційною системою. Вона дозволяє виконувати команди, управляти процесами, створювати і видаляти файли та каталоги, однак не містить мовних засобів для розв'язання інших задач, таких як обробка даних або створення програмного забезпечення.

Під час вибору мови програмування для розв'язання конкретної задачі слід враховувати такі фактори:

- тип задачі, яку потрібно розв'язати;
- сфера застосування мови програмування;
- розмір і складність програми;
- особливі вимоги до програми.

## МОВИ ПРОГРАМУВАННЯ НИЗЬКОГО І ВИСОКОГО РІВНІВ

Мови програмування розвивалися від простих мов низького рівня, що безпосередньо керували процесором, до складних мов високого рівня, які є більш абстрактними і зрозумілими людям (табл. 1.3).

**Таблиця 1.3** – Основні характеристики мов програмування різного рівня

| Рівень | Характеристика | Переваги / недоліки | Приклад | Автор | Роки |
|---|---|---|---|---|---|
| Низький | Описують програми в термінах машинних інструкцій | Ефективність, повний контроль / Складність, низька інтегрованість, обмежені можливості | Машинний код, асемблер | Розробники перших комп'ютерів | 1940-ві |
| Високий | Описують програми в термінах понять, більш зрозумілих людям | Простота, інтегрованість, широкі можливості / Неефективність, відсутність повного контролю | C, C++, Java, Python, JavaScript | Джон Бекус, Конрад Цузе, Джон Кемені, Том Курц, Денніс Річі, Ніклаус Вірт, Бьярн Страуструп, Джеймс Гослінг | 1950-ті – 2000-ні |

Перші мови програмування були мовами низького рівня, які безпосередньо керували процесором. Вони були розроблені в 1940-х роках, коли комп'ютери були ще дуже простими і мали обмежений набір можливостей.

Одною із перших мов низького рівня був машинний код, який являє собою послідовність бітів, що безпосередньо інтерпретуються процесором. Машиний код дуже складний для написання та розуміння, тому його використовують лише для найпростіших завдань.

Для спрощення цієї задачі почали з'являтися мови програмування низького рівня, які дозволяли задавати машинні команди в більш зрозумілому для людини вигляді (рис. 1.2).

Іншою ранньою мовою низького рівня був асемблер, який використовує мнемонічні коди для позначення машинних інструкцій. Він трохи простіший для написання, ніж машинний код, але все одно вимагає від програміста глибокого розуміння роботи процесора.

У 1950-х роках почалася розробка мов високого рівня, які були більш абстрактними і зрозумілими людям. Першою мовою високого рівня була Планкалкюль, розроблена Конрадом Цузе в 1943–1945 рр. Однак вона не мала комп'ютерної реалізації і не набула широкого поширення.

Першою мовою високого рівня, що набула широкого поширення, став Фортран, розроблений у 1954 р. групою Джона Бекуса. Він був призначений для наукових обчислень і мав ряд переваг перед мовами машинного коду, зокрема, був більш простим у використанні і дозволяв використовувати математичні функції.

| Машинний код | | Асемблер | |
|---|---|---|---|
| 0005 | B4 09 | 7 | mov AH, 09 h |
| 0007 | BA0000r | 8 | mov DX, offset msg |
| 00A | CD 21 | 9 | int 21 h |

**Рисунок 1.2** – Приклад машинного коду і представлення його на асемблері

## Мови програмування низького рівня

Мови програмування низького рівня, такі як асемблер, використовують мнемонічні коди для позначення машинних інструкцій, які більш зрозумілі для людини, ніж двійкові коди, що полегшує написання і розуміння програм. Для перетворення їх у двійковий код було створено спеціальні програми – транслятори.

**Мова машинного коду** – це набір двійкових кодів, які безпосередньо керують процесором. Двійкові коди дуже складні для розуміння та написання, тому вони не є ефективним способом програмування для людей.

**Мова програмування низького рівня** – це мова програмування, яка використовує мнемонічні коди для позначення машинних інструкцій. Мнемонічні коди є більш зрозумілими для людини, ніж двійкові коди, тому вони полегшують написання і розуміння програм.

**Транслятор** – це програма, яка перетворює програми, написані на одній мові програмування, у програми, написані на іншій мові програмування. У випадку мов програмування низького рівня транслятори перетворюють програми, написані на мові програмування низького рівня, у двійковий код, який може бути безпосередньо виконаний процесором (рис. 1.3).

Транслятори поділяють на два типи:

**Компілятор** – транслятор, який перетворює програму повністю, до її виконання. Він створює об'єктний файл, який містить двійковий код програми. Для виконання програми об'єктний файл повинен бути зв'язаний з бібліотеками, які містять необхідні функції і дані.

**Інтерпретатор** – транслятор, який перетворює програму послідовно, під час її виконання. Він не створює об'єктний файл, а перетворює код програми безпосередньо в машинний код, який потім виконується процесором.

*Переваги компіляторів:*

Ефективність: програми, згенеровані компілятором, зазвичай більш ефективні, ніж програми інтерпретовані. Це пов'язано з тим, що компілятор може оптимізувати код програми для конкретного процесора.

Переносимість: програми, згенеровані компілятором, часто більш переносимі, ніж програми, інтерпретовані. Це пов'язано з тим, що компілятор створює об'єктний файл, який не залежить від конкретної архітектури комп'ютера.

*Недоліки компіляторів:*

Швидкість виконання: програми, згенеровані компілятором, зазвичай виконуються повільніше, ніж програми інтерпретовані. Це пов'язано з тим, що компілятор повинен спочатку створити об'єктний файл, а потім виконати його.

Розмір програми: програми, згенеровані компілятором, зазвичай мають більший розмір, ніж програми інтерпретовані. Це пов'язано з тим, що компілятор повинен включати в об'єктний файл інформацію про типи даних, змінні та функції програми.

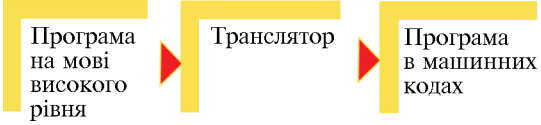



**Рисунок 1.3.** – Схематичне представлення роботи транслятора

*Переваги інтерпретаторів:*

Швидкість виконання: програми інтерпретовані зазвичай виконуються швидше, ніж програми, згенеровані компілятором. Це пов'язано з тим, що інтерпретатор не повинен спочатку створювати об'єктний файл, а просто перекладає код програми безпосередньо в машинний код.

Розмір програми: програми інтерпретовані, зазвичай мають менший розмір, ніж програми, згенеровані компілятором. Це пов'язано з тим, що інтерпретатор не повинен включати в програму інформацію про типи даних, змінні та функції.

*Недоліки інтерпретаторів:*

Неефективність: програми інтерпретовані зазвичай менш ефективні, ніж програми, згенеровані компілятором. Це пов'язано з тим, що інтерпретатор повинен перекладати код програми послідовно, під час її виконання.

Вибір типу транслятора залежить від конкретних вимог до програми. Якщо важлива ефективність програми, то краще використовувати компілятор. Якщо важлива швидкість виконання програми, то краще використовувати інтерпретатор. Якщо важлива переносимість програми, то краще використовувати компілятор.

*Приклади трансляторів:*

Компілятори: GCC, Clang, Visual Studio Compiler, Xcode Compiler.

Інтерпретатори: Python Interpreter, Ruby Interpreter, JavaScript Interpreter.

## Мови програмування високого рівня

Перші мови програмування високого рівня були створені в 1950-х роках. Вони були розроблені для того, щоб полегшити процес програмування для людей. До перших мов високого рівня відносять Fortran, Lisp і Cobol.

Fortran було розроблено у 1954 р. для наукових розрахунків. Він був першою мовою програмування, яка використовувала мнемонічні коди для позначення машинних інструкцій.

Lisp розроблено у 1958 р. для символьних обчислень. Він був першою мовою програмування, яка використовувала рекурсію.

Cobol розроблено в 1959 р. для бізнес-додатків. Він був першою мовою програмування, яка використовувала структурований підхід до програмування.

У наступні роки було розроблено нові мови програмування високого рівня, такі як Pascal, C, Java, Python тощо. Ці мови програмування значно полегшили процес програмування і зробили його більш доступним для людей.

Мови програмування високого рівня використовують такі принципи:

*Абстракція:* мови абстрагують від деталей апаратного забезпечення і операційної системи. Це дозволяє програмістам зосередитися на логіці програми, а не на її реалізації.

*Інструментальність:* мови програмування високого рівня забезпечують програмістів інструментами, які допомагають їм писати програми, наприклад, компіляторами, інтерпретаторами та відладувачами.


**Рисунок 1.4** – Мови програмування високого рівня

*Класифікація:* мови програмування високого рівня класифікують за різними ознаками, наприклад, за призначенням, за типом даних, за парадигмою програмування.

*Переваги мов програмування високого рівня:*

Простота: мови програмування високого рівня використовують такі поняття, як оператори, цикли, підпрограми та структури даних, які є зрозумілими для людей.

Переносимість: програми, написані на мовах програмування високого рівня, часто є переносимими, тобто можуть виконуватися на різних платформах.

Ефективність: програми, написані на мовах програмування високого рівня, часто можуть бути виконані так само ефективно, як і програми, написані на мовах програмування низького рівня.

*Недоліки мов програмування високого рівня:*

Ефект інверсії контролю: мови програмування високого рівня використовують такі поняття, як цикли, підпрограми та структури даних, які можуть призвести до ефекту інверсії контролю, коли програміст не контролює потік виконання програми.

Штучність: мови програмування високого рівня використовують такі поняття, як оператори, цикли, підпрограми та структури даних, які можуть бути штучними для деяких програмістів.

За останні 5 років найбільш поширеними мовами програмування були такі:

Python: Python є всебічною мовою програмування, яка використовується для різних завдань, таких як веб-розробка, машинне навчання і наука про дані.

JavaScript: JavaScript є мовою програмування, яка використовується для створення інтерактивних веб-додатків.

Java: Java є мовою програмування, яка використовується для розробки мобільних додатків, веб-додатків і корпоративних додатків.

C++:C++ є мовою програмування, яка використовується для розробки системного програмного забезпечення, ігор і 3D-графіки.

C: C є мовою програмування, яка використовується для розробки системного програмного забезпечення, ігор і 3D-графіки.

Ці мови програмування є популярними завдяки простоті, переносимості, ефективності і можливості (рис. 1.4).

## ІСТОРІЯ РОЗВИТКУ МОВ ПРОГРАМУВАННЯ

Історія створення першої мови програмування бере свій початок у 1943 р., коли німецький інженер Конрад Цузе розробив мову програмування під назвою Plankalkül. Планкалькюль був першою мовою програмування високого

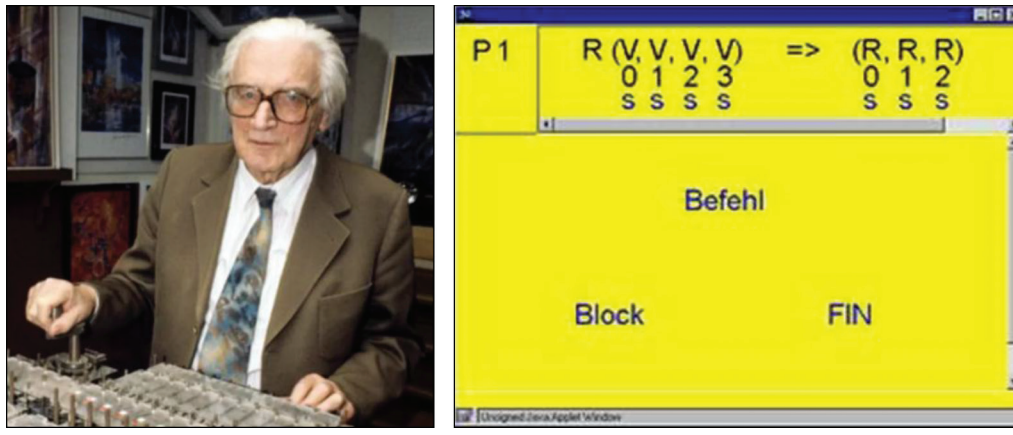



**Рисунок 1.5** – Конрад Цузе (1910–1955) та частина машинного коду на мові Планкалькюль

рівня, яка використовувала англоподібні ключові слова та синтаксис. Однак, у той час він не був реалізований і не мав значного впливу на розвиток мов програмування (рис. 1.5).

У 1954 р. Джон Бекус, програміст IBM, розробив мову програмування під назвою FORTRAN. Це була перша мова програмування високого рівня, яка набула широкого поширення та використовувалася для наукових обчислень. FORTRAN був написаний на машинній мові, але він використовував англоподібні ключові слова та синтаксис, що робило його набагато простішим у вивченні та використанні, ніж машинні мови (рис. 1.6).

У 1958 р. Джон Маккарті, професор Массачуссетського технологічного інституту, розробив мову програмування під назвою LISP. Це була перша мова програмування, яка використовувала рекурсію, що є потужною технікою для вирішення проблем. LISP став основою для розвитку об'єктно-орієнтованого програмування та штучного інтелекту.

У 1959 р. Грейс Хоппер, програмістка з ВМС США, розробила мову програмування під назвою COBOL, що була першою мовою програмування, котру використовували для бізнес-додатків. COBOL був написаний на машинній мові, але використовував англоподібні ключові слова та синтаксис,

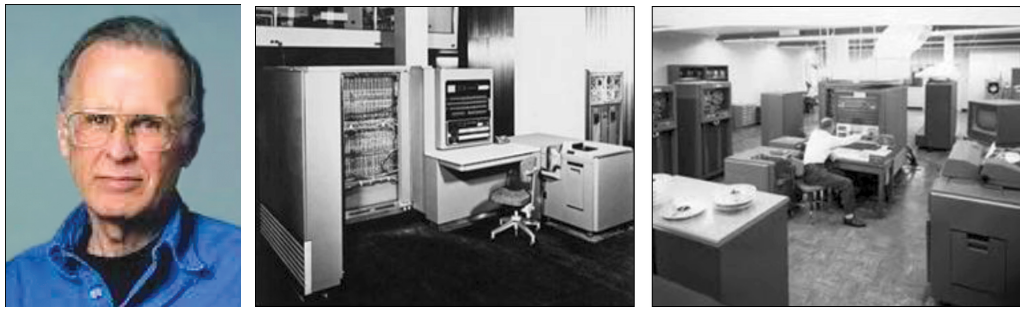
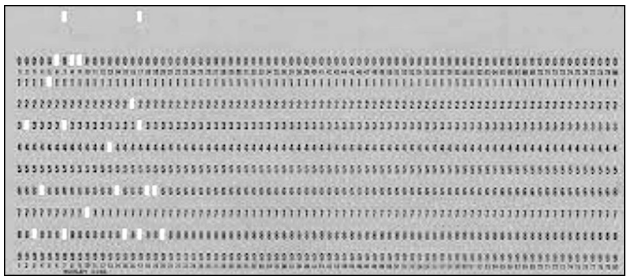


**Рисунок 1.6** – Джон Бекус (1924–2007); IBM 701; IBM 704



Фортран IV

Фортран 77

**Рисунок 1.7** – Мови програмування

що робило його набагато простішим у вивченні та використанні, ніж машинні мови.

Ці три мови програмування – FORTRAN, LISP і COBOL – стали основою для розвитку сучасних мов програмування. Вони зробили програмування простішим і ефективнішим, що привело до бурхливого розвитку галузі програмного забезпечення.

Першою широковживаною компільованою мовою став *FORTRAN,* розроблений групою Джона Бекуса, анонсований у 1954 р. і випущений у 1957 р. для IBM 704 (рис. 1.7).

Головне призначення мови FORTRAN – це наукові обчислення, для вирішення складних математичних та наукових задач.

Використання мови FORTRAN для наукових обчислень бере свій початок у 1950-х роках. Вона мала ряд функцій, які робили її ідеальною для цієї мети, включаючи підтримку математичних операцій, складних структур даних і паралельного програмування. FORTRAN швидко став стандартом для наукових обчислень. Він був використаний для розробки багатьох важливих наукових програм, включаючи програми для прогнозування погоди, моделювання клімату, дослідження природних явищ і розробки ядерної зброї.

У 1970–1980-х роках FORTRAN продовжував залишатися популярною мовою для наукових обчислень. Однак, у цей період також було розроблено нові мови програмування, які були більш потужними та гнучкими, ніж FORTRAN. Наприклад, мова C була розроблена для системного програмування, але вона також мала широкий спектр функцій, які робили її придатною для наукових обчислень.

У 1990–2000-х роках FORTRAN почав поступово втрачати популярність. Однак, він досі залишається популярною мовою для наукових обчислень у деяких галузях, таких як аерокосмічна інженерія та ядерна фізика.

FORTRAN – це потужна мова програмування, яка має широкий спектр застосувань у наукових обчисленнях і, ймовірно, буде використовуватися ще багато років.

*LISP* – скорочення від List Processing – це мова програмування високого рівня, яка була розроблена в 1958 р. Джоном Маккарті – професором Массачуссетського технологічного інституту. LISP була першою мовою програмування, яка використовувала рекурсію, що є потужною технікою для вирішення

проблем. Вона була розроблена для символьних обчислень, таких як обробка природних мов, машинний переклад і штучний інтелект. Він швидко став популярною мовою для цих застосувань.

У 1960-х роках було розроблено кілька діалектів LISP, включаючи Common Lisp і Scheme. Common Lisp став стандартом для LISP і досі є найпоширенішим діалектом.

У 1970–1980-х роках LISP був використаний для розробки багатьох важливих систем штучного інтелекту, включаючи експертні системи, розпізнавання образів і штучне бачення.

У 1990–2000-х роках LISP продовжував залишатися популярною мовою для штучного інтелекту, але також став популярним для інших застосувань, таких як веб-розробка і науки про дані. Це і досі потужна і гнучка мова програмування, яка має широкий спектр застосувань. Вона є основою для багатьох важливих систем штучного інтелекту і сьогодні продовжує залишатися популярною мовою для програмування.

У 1958 р. Джон Бекус, програміст IBM, запропонував розробити універсальну мову програмування, яка могла б бути використана для будь-яких цілей. Він запропонував назву *ALGOL*, що є абревіатурою від Algorithmic Language.

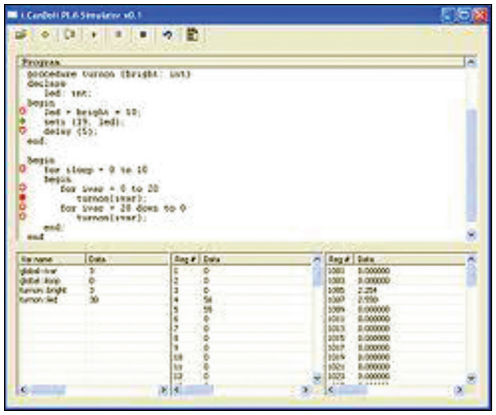
Мова була розроблена групою вчених під керівництвом Джона Бейкуса. Вона була представлена в 1960 р. і стала першою широко використовуваною універсальною мовою програмування. ALGOL стала основою для багатьох інших мов програмування, включаючи C, C++ і Java. Її досі використовують у деяких галузях, таких як наукові обчислення і штучний інтелект.

*JOVIAL* – ця мова програмування була розроблена в 1960 р. для використання в системах управління ракетами і космічними апаратами. JOVIAL – абревіатура від Joint Optimizing VAriational Language. Вона була розроблена групою вчених під керівництвом Роберта Макдональда з Військово-повітряних сил США. Розробка мови була розпочата в 1958 р. і завершилася в 1960 р.

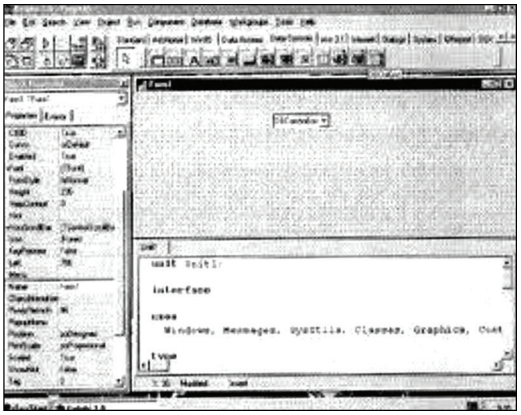
Її розробляли для того, щоб забезпечити універсальну мову програмування, яка могла б бути використана для розробки широкого спектра програмного забезпечення для систем управління ракетами і космічними апаратами. JOVIAL була покликана замінити ряд існуючих мов програмування, які використовували для цих цілей. Її використовували для розробки програмного забезпечення для таких програм, як:

- Програма Аполлон – для розробки програмного забезпечення для космічних кораблів Аполлон, які доставили людей на Місяць.
- Програма Space Shuttle – для розробки програмного забезпечення для космічного корабля Space Shuttle.
- Програма GPS – для розробки програмного забезпечення для глобальної системи позиціонування (GPS).

Цю мову продовжували використовувати в системах управління ракетами і космічними апаратами до кінця 1980-х років. Однак, у 1990-х роках вона була замінена іншими мовами програмування, такими як Ada і C++. Це потужна і гнучка мова програмування, яка мала важливий вплив на розвиток систем управління ракетами і космічними апаратами.



BASIC



SIMULA 67

**Рисунок 1.8** – Мови програмування

*COBOL* – мова програмування, яка вплинула на розвиток бізнес-додатків. COBOL (скорочення від COmmon Business Oriented Language) – це мова програмування високого рівня, яка була розроблена в 1959 р. для використання в бізнес-додатках. Її було розроблено групою вчених під керівництвом Дональда Весткотта з Комітету з мов для обробки даних (CODASYL). Розробка була розпочата в 1959 р. і завершилася в 1960 р. Створена для того, щоб забезпечити універсальну мову програмування, яка могла б бути використана для розробки широкого спектра бізнес-додатків. COBOL була широко використана в бізнес–додатках. Вона використовувалася для розробки програмного забезпечення таких програм: банківські системи – для банківських систем, які обробляють фінансові транзакції та системи управління запасами; для систем управління запасами, які відстежують запаси товарів.

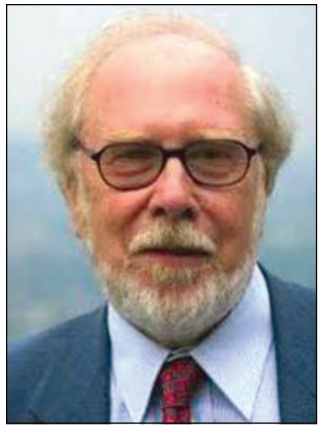
*SIMULA 67* – мова програмування, розроблена в 1965 р. для моделювання систем, які складаються із взаємодіючих об'єктів. SIMULA 67 – це абревіатура від Simulation Language 67.

Її було розроблено групою вчених під керівництвом Олафа Нюгорд–Рангена з Норвезького університету в Тронгеймі. Розробка мови була розпочата в 1962 р. і завершилася в 1967 р. (рис. 1.8).

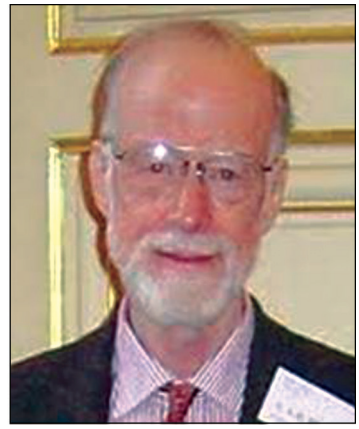
Мову було розроблено для того, щоб забезпечити мову програмування, яка могла б бути використана для моделювання широкого спектра систем, включаючи фізичні, біологічні та соціальні системи. Вона була покликана замінити ряд існуючих мов програмування, які використовувалися для цих цілей.

Мова *BASIC* була створена в 1963 р. професорами Дартмутського коледжу Джоном Кемені і Томасом Курцом. Назва є акронімом від фрази Beginner's All–purpose Symbolic Instruction Code, що означає «універсальний код символьних інструкцій для початківців».

Учені хотіли створити мову програмування, яка була б простою у вивченні та використанні і при цьому дозволяла б програмістам створювати ефективні програми. Вони вважали, що мови програмування, які були поширені на той час, занадто складні для початківців.



Ніклаус Вірт



Тоні Хоар

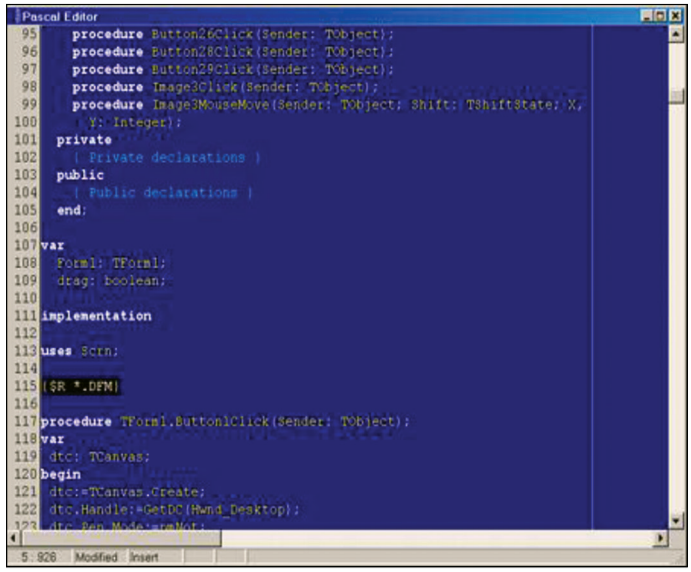



**Рисунок 1.9** – Розробники мов програмування Pascal

Мова BASIC (див. рис. 1.8) мала значний вплив на розвиток програмування. Вона була першою мовою програмування, розробленою спеціально для початківців, і зробила програмування доступнішим для широкого кола людей. Також була основою для багатьох інших мов програмування, які були розроблені пізніше. Вона мала ряд функцій, які робили її придатною для цієї мети.

BASIC швидко стала популярною мовою програмування серед початківців. Її використовували для навчання програмування в школах і університетах, а також для розробки простих програм, таких як ігри, навчальні програми та програми для роботи з даними.

У 1970-х роках BASIC була стандартизована організацією ANSI. Це дозволило забезпечити сумісність різних версій мови. У 1980-х роках вона стала

популярною мовою програмування для персональних комп'ютерів. Її використовували для розробки програм таких операційних систем, як MS–DOS і Apple DOS. У 1990-х роках BASIC почала поступатися місцем іншим мовам програмування, таким як C і Java, однак, все ще залишається популярною мовою програмування для початківців.

Мова *Паскаль* була створена в 1970 р. швейцарським математиком Ніклаусом Віртом. Назва є даниною поваги французькому філософу і математику Блезу Паскалю, якого вважають одним із засновників сучасної інформатики (рис. 1.9)

Учений хотів створити мову програмування, яка була б простою у вивченні та використанні і при цьому дозволяла б програмістам створювати ефективні програми. Він вважав, що мови програмування, які були поширені на той час, були занадто складними і заплутаними.

Паскаль була першою мовою програмування, розробленою для навчання програмування. Її використовували в школах і університетах в усьому світі, а також для розробки широкого спектра програм, включаючи наукові програми, бізнес–додатки та ігри. У 1980-х роках Паскаль була стандартизована організацією ISO. Це дозволило забезпечити сумісність різних версій мови.

У 1990-х роках Паскаль почала поступатися місцем іншим мовам програмування, таким як C++ і Java, проте все ще залишається популярною мовою програмування для навчання програмування. Паскаль – це мова високого рівня, а це означає, що вона використовує англоподібні ключові слова та синтаксис. Це робить її набагато простішою у вивченні та використанні, ніж машинні мови, які були поширені на той час.

Мова Паскаль мала значний вплив на розвиток програмування. Вона була першою мовою програмування, яка була розроблена спеціально для навчання програмуванню, і вона зробила програмування доступнішим для широкого кола людей. Паскаль також була основою для багатьох інших мов програмування, які були розроблені пізніше (див. рис. 1.9).

Мова *C* була створена в 1972 р. Деннісом Рітчі в Bell Telephone Laboratories з метою написання операційної системи UNIX. Назва є абревіатурою від. System Programming Language, що означає «мова програмування для системного програмування» (рис. 1.10).

Учений хотів створити мову програмування, яка була б простою й ефективною і при цьому дозволяла б програмістам писати складне програмне забезпечення. Він вважав, що мови програмування, які були поширені на той час, занадто складні і заплутані.

C була першою мовою програмування, розробленою спеціально для системного програмування. Вона мала ряд функцій, які робили її придатною для цієї мети, тому швидко стала популярною. Її використовували для розробки операційних систем, компіляторів, бібліотек і інших системних програм. У 1980-х роках C стала популярною мовою програмування для широкого спектра програм, включаючи наукові програми, бізнес–додатки та ігри. У 1990-х роках C почала поступатися місцем іншим мовам програмування, таким як C++ і Java, однак, все ще залишається популярною мовою програмування для системного програмування.

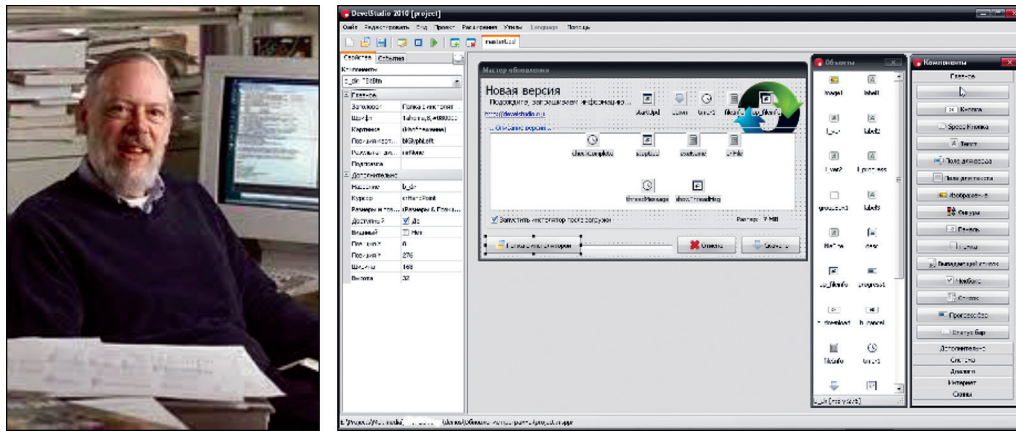


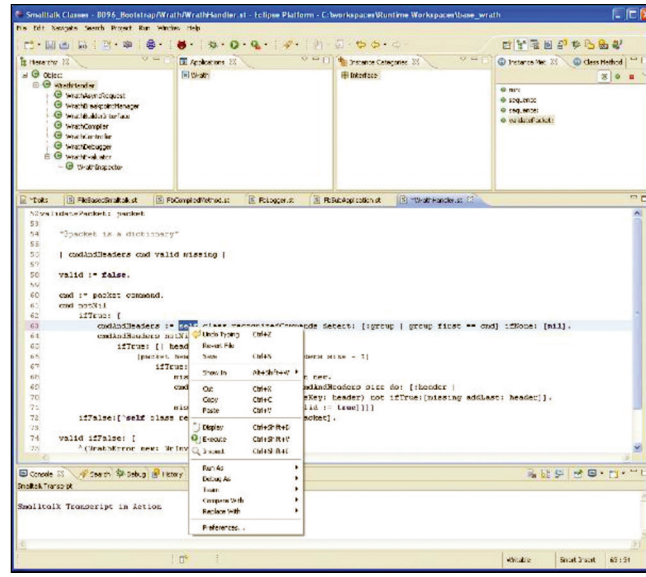
**Рисунок 1.10** – Розробник мови програмування С Денніс Річі

Мова С мала значний вплив на розвиток програмування, оскільки зробила програмування операційних систем і інших системних програм доступнішим для широкого кола людей. Також була основою для багатьох інших мов програмування, які були розроблені пізніше, таких як C++, Java і Python.

У той самий час у Марселі було створено інтерпретатор мови *Пролог* – першу і найвідомішу мову логічного програмування. Мова логічного програмування на відміну від процедурних мов, за допомогою яких вирішується питання «ЯК» робити, зосереджена на тому «ЩО» робити. Тобто програміст повідомляє, що йому відомо й задає питання, а система сама будує дедуктивний висновок.

Мова *Smalltalk* була створена в 1972 р. Аланом Кейєм, Деном Інгаллсом, Тедом Кеглером і Адель Голдберг в Xerox PARC. Назва мови є абревіатурою від англ. Smalltalk-72, що означає «маленький розмовний діалог 1972 року» (рис. 1.11).

**Рисунок 1.11** – Розробник мови програмування Smalltalk Алан Кей



Розробники хотіли створити мову програмування, яка була б простою й ефективною і при цьому дозволяла б програмістам писати програмне забезпечення нового покоління, яке б було орієнтоване на взаємодію з користувачем. Вони вважали, що мови програмування, які були поширені на той час, занадто складні і заплутані.

Smalltalk швидко стала популярною мовою програмування для об'єктно–орієнтованого програмування. Вона використовувалася для розробки інтерфейсів користувача, баз даних, систем управління та інших програм.

У 1980-х роках вона стала популярною мовою програмування для навчальних цілей, використовувалася в школах і університетах в усьому світі.

У 1990-х роках Smalltalk почала поступатися місцем іншим мовам програмування, таким як Java і C#, однак, все ще залишається популярною мовою програмування для об'єктно-орієнтованого програмування.

Мова Smalltalk мала значний вплив на розвиток програмування, оскільки зробила об'єктно-орієнтоване програмування доступнішим для широкого кола людей. Також стала основою для багатьох інших мов програмування, які були розроблені пізніше, таких як Objective–C, C++, Java і Python.

Мова *ML* була розроблена в Единбурзькому університеті Робіном Мілнером у 1973 р. (рис. 1.12). Мілнер був стурбований тим, що існуючі мови програмування не підходили для обробки символьних даних, таких як теорія типів, мова природних мов і штучний інтелект. Він хотів створити мову, яка була б ефективною для таких завдань, але при цьому залишалася б простою у використанні, тому почав розробляти ML, спираючись на досвід роботи з мовою LISP. Він запропонував ряд нових мовних конструкцій, які були призначені для полегшення обробки символьних даних, наприклад, ввів типи даних для символів, рядків і списків, також ввів оператори для роботи з цими типами даних, такими як оператори конкатенації та розщеплення.

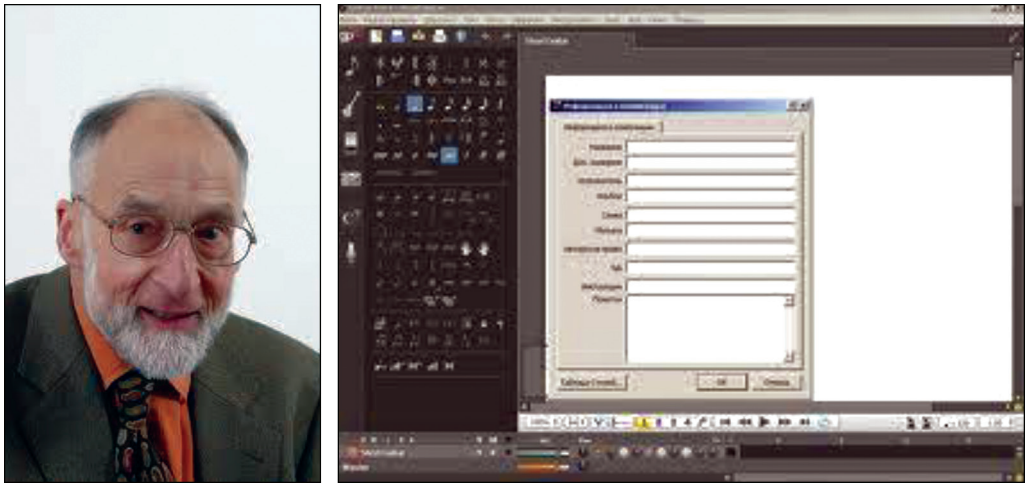


**Рисунок 1.12** – Розробник мови програмування ML Робін Мілнер

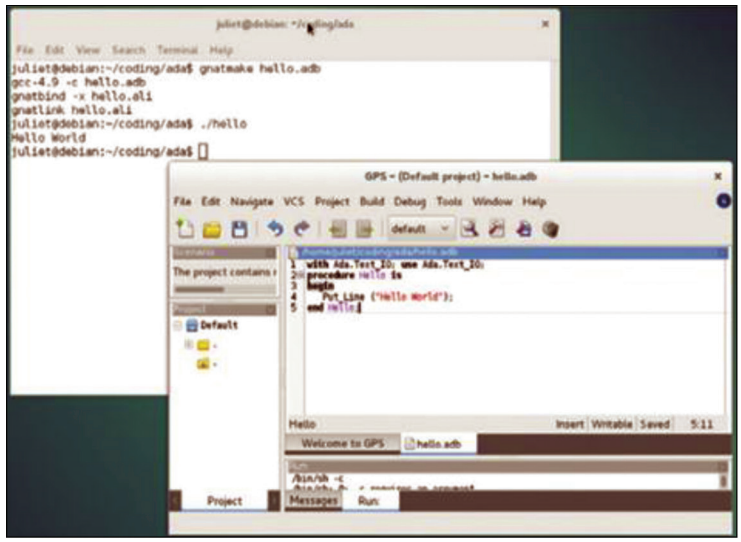
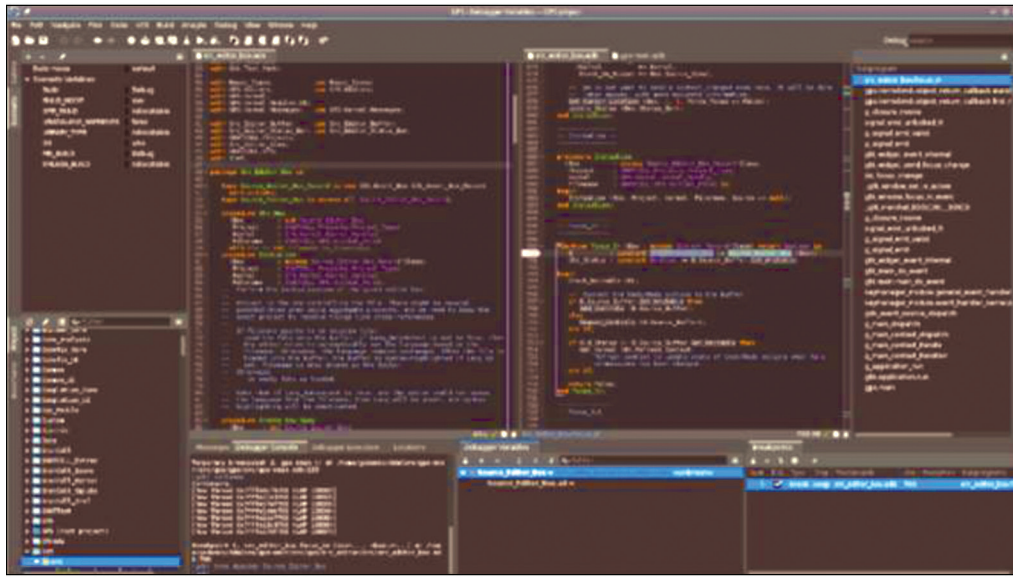



**Рисунок 1.13** – Мова програмування Ада



Програмування в Ада сьогодні

ML була першою мовою програмування, розробленою спеціально для обробки символьних даних. Вона швидко стала популярною серед дослідників у галузі штучного інтелекту та обробки природної мови і використовується сьогодні в цих галузях, а також в інших сферах, таких як математика, логіка та теорія типів.

У 1975 р. в Массачуссетському технологічному інституті описано спрощений діалект мови – Scheme.

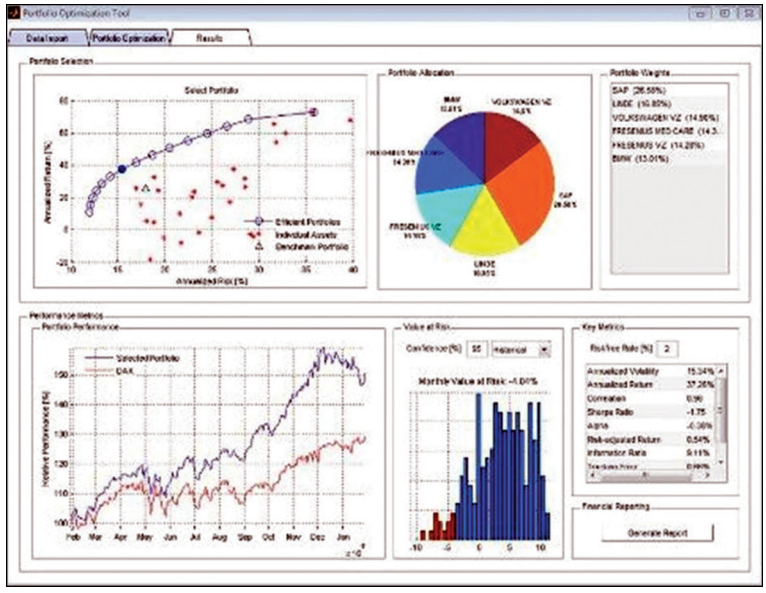


**Рисунок 1.14** – Програмування у програмі MATLAB

Мова *Ада*. (рис. 1.13) була створена у 1977–1983 рр. групою вчених під керівництвом Жана Ішбіа (Jean Ichbiah) на фірмі Сіі Honeywell Bull за ініціативи Міністерства оборони США.

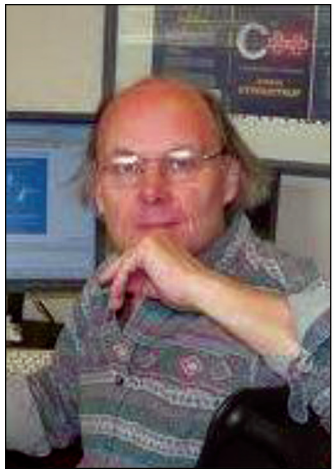
Мета створення мови Ада полягала в тому, щоб розробити універсальну мову програмування, яка була б безпечною, надійною та ефективною для критичних до безперервності систем. Учений був стурбований тим, що існуючі мови програмування не відповідали вимогам Міністерству оборони США до мов програмування для критичних до безперервності систем, які часто були складними та ненадійними, що могло призвести до помилок у програмуванні, які могли б мати серйозні наслідки.

У 1984 р. з метою об'єднання різних діалектів LISP створено Common Lisp. Випущено *MATLAB* (рис. 1.14).

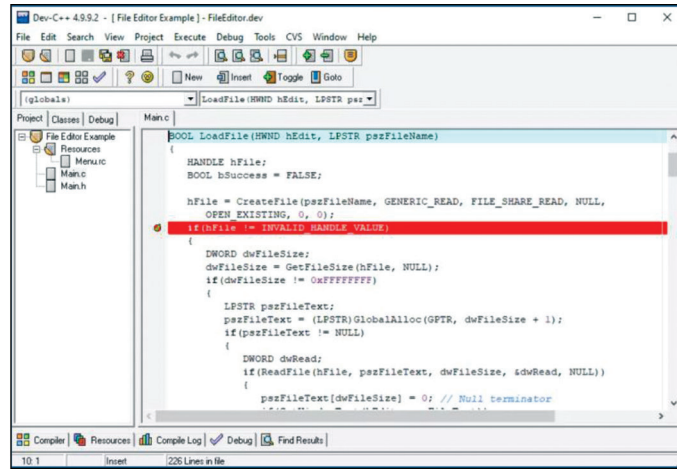
Мова *C++* була розроблена Б'єрном Страуструпом (Bjarne Stroustrup) у Bell Labs. Страуструп був стурбований тим, що мова С, яка була тоді популярною мовою програмування для системного програмування, не була достатньою для потреб сучасних операційних систем. Він хотів створити мову, яка була б сумісна з С, але при цьому мала б ряд додаткових функцій, таких як об'єктно-орієнтоване програмування та управління пам'яттю. Він почав розробляти С++, спираючись на досвід роботи з мовою С, і вніс ряд змін до мови С, щоб зробити її більш об'єктно-орієнтованою та ефективною.

Деякі з основних характеристик мови С++:

• Об'єктно-орієнтоване програмування: підтримує об'єктно-орієнтоване програмування, що дозволяє розробникам створювати складні програми з повторюваними компонентами.



Б’ярн Страуструп




Одна з найкращих IDE для C++ Dev–C++

**Рисунок 1.15** – Мови програмування

• Управління пам’яттю: має функції для управління пам’яттю, що дозволяє розробникам уникати помилок, пов’язаних із пам’яттю.

• Сумісність з C: сумісна з C, що дозволяє розробникам використовувати код, написаний на C, у програмах на C++.

Мова C++ була випущена в 1985 р. Вона швидко стала популярною серед розробників операційних систем, а також серед розробників інших програм.

На сьогодні C++ є однією з найпопулярніших мов програмування. Її використовують для розробки широкого спектра програм, включаючи операційні системи, драйвери пристроїв, веб-додатки та ігри.

Деякі з прикладів використання мови C++:

• Операційні системи, такі як Linux, Windows та macOS.

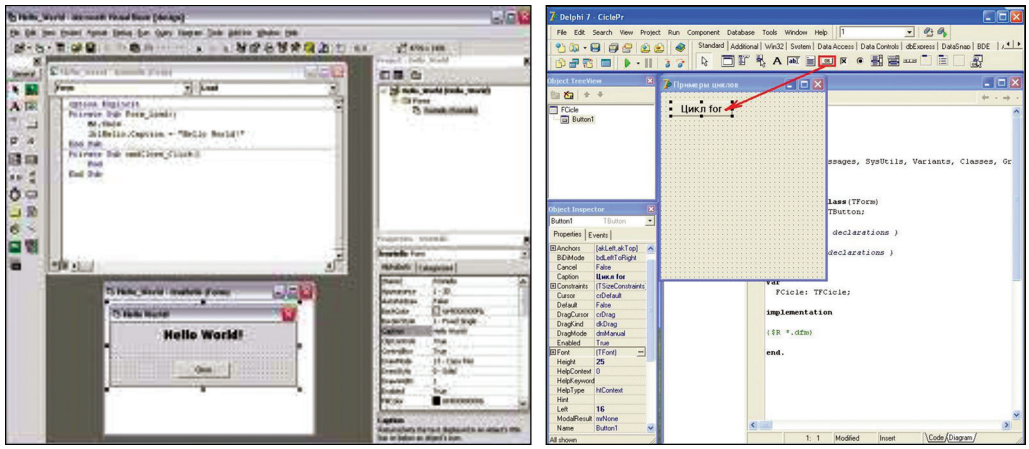
• Драйвери пристроїв, такі як драйвери для графічного процесора, мережевої карти та принтера.

• Веб-додатки, такі як WordPress, Joomla та Drupal.

• Ігри, такі як Doom, Quake та Minecraft.

У 1985 р. Б’ярн Страуструп опублікував реалізацію мови C++. Ця мова до сьогодні залишається однією із найпопулярніших мов програмування: її використовують понад 4,5 млн розробників. Зазначимо, що створення інтегрованого середовища розробки (IDE) у 1989 р. на основі операційної системи UNIX спростило процес компіляції, налагодження та виконання коду програм. Сучасні IDE містять щонайменше один редактор вихідного коду, відладчик та різні інструменти автоматизації. Деякі з них інтегровані з компілятором, інтерпретатором та розширеними функціями, такими як автоматичне завершення коду, збирання та розгортання (рис. 1.15).

У 1986 р. опубліковано мову Objective-C і створено Erlang. Тоді ж Borland і Apple незалежно створили об’єктно-орієнтоване розширення мови Pascal – *Object Pascal*. Ця мова більш відома як основна мова програмування середовища Delphi. А безпосереднім попередником Object Pascal була Turbo Pascal.




Програмування на Object Pascal в середовищі Delphi 7

**Рисунок 1.16** – Мови програмування

У 1987 р. створено *Perl.*

У 1990 р. опубліковано *Standard ML* і *Haskell.*

У 1991 р. створено *Visual Basic* і опубліковано *Python*. Щодо Visual Basic, то її розроблено на основі синтаксису QuickBasic компанією Microsoft для своєї операційної системи Windows (рис. 1.16).

Головною перевагою програми став зв'язок з графічним інтерфейсом, раніше реалізований програмістом Аланом Купером у прототипі Tripod. Але саме Visual Basic 1.0 ця реалізація досягла потрібного рівня. А мова Python увібрала в себе багато ідей з ABC, а також з інших мов, наприклад, система модулів взята з мови Modula-3. Нині Python обирають для навчання програмування початківців.

У 1992 р. випущено *Oracle 7* з підтримкою PL/SQL.

У 1993 р. створено *Lua.*

У 1995 р. Sun Microsystems випустила *Java*, Netscape — *JavaScript*, тоді ж створено *PHP* і *Ruby.*

Зауважимо, що програмування на Java вважається більш динамічним, ніж на C чи C++, оскільки мова призначена для адаптації до мінливих умов, на ній можна писати програми, спрямовані на виконання багато задач одночасно.

JavaScript – одна із найбільш популярних мов програмування. Схожість між Java і JavaScript передусім за синтаксисом. Але насправді це дуже різні інструменти. Якщо Java є об'єктно-орієнтованою мовою програмування, то JavaScript – це об'єктно-орієнтована мова створення сценаріїв. Водночас Java використовується для створення програм, які запускаються на віртуальних машинах або браузерах, а код JavaScript виконується тільки в браузері. Крім того, Java переважно використовується при написанні серверних програм, а JavaScript – для створення інтерактивних веб-сторінок та додатків (рис. 1.17).

У 1996 р. створено *OCaml.*

У 2001 р. створено *C#* (Сі Шарп). Це одна з найбільш затребуваних і при цьому «зручних» мов програмування, які швидко розвиваються. C# являє со-

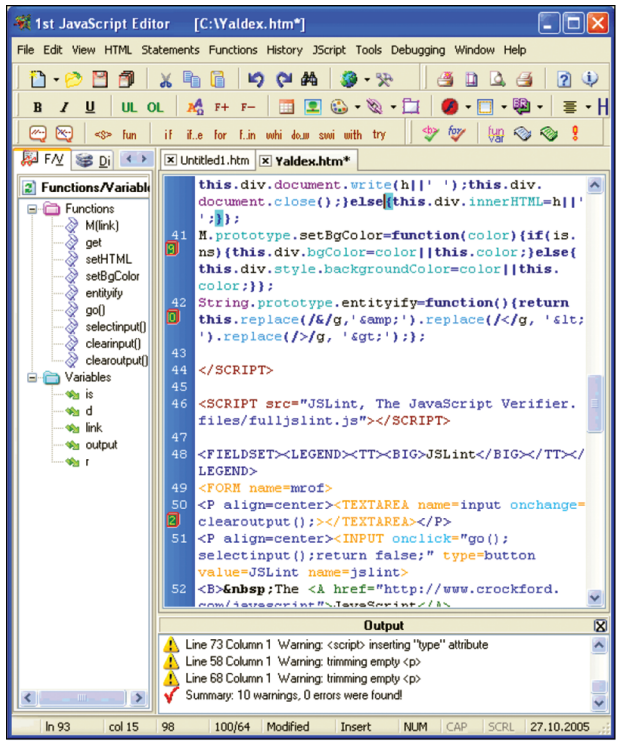
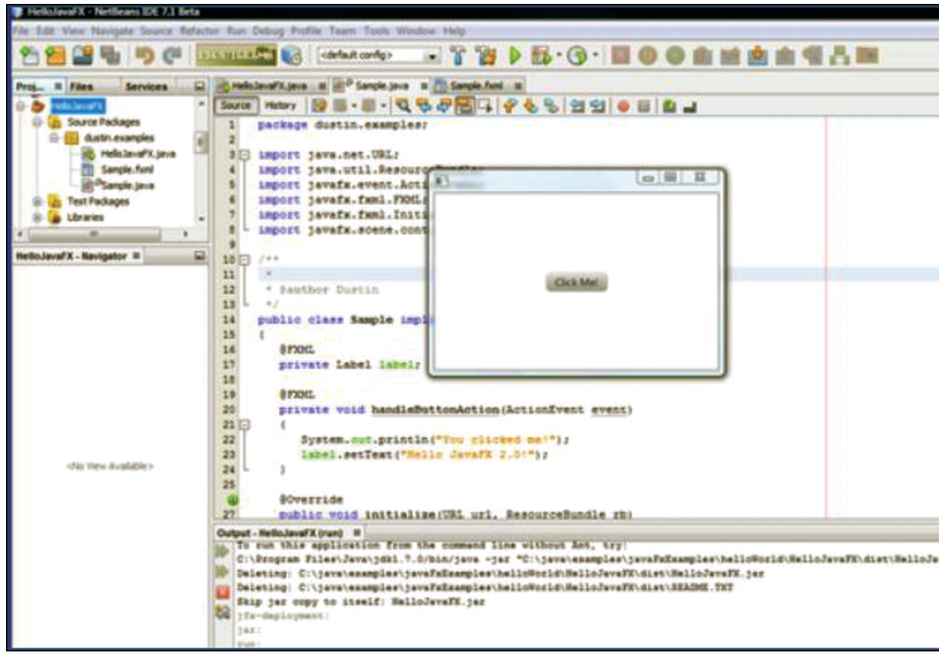
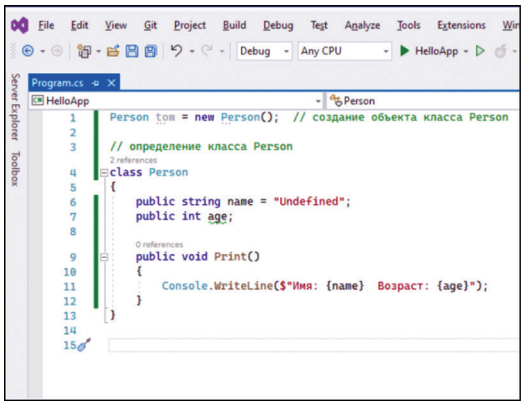


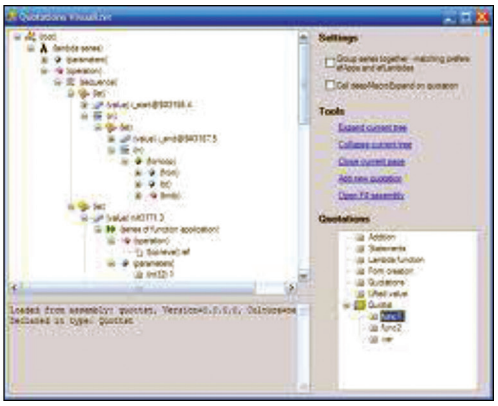

**Рисунок 1.17** – Мови програмування

JavaScript



Java




C# F#

**Рисунок 1.18** – Мови програмування

бою модифікацію фундаментальної мови C від компанії Microsoft. Мова дозволяє створити найбільш універсальний засіб для розробки програмного забезпечення для великої кількості пристроїв і операційних систем. Синтаксис мови наближений до Java та C++.

У 2002 р. створено *F#*. Якщо C# більше підходить для розробки інтерфейсів, роботи з графікою або написання логіки, в якій потрібно працювати з об'єктами, то F# краще використовувати для написання бізнес-логіки, яка потребує серйозних обчислень. Зазначимо, що серед розробників ігор популярніший C# (рис. 1.18).

У 2003 р. на основі Java створено мову програмування *Scala*. Але вона не отримала такого поширення, як попередниця. Незважаючи на більш чітку й лаконічну мову, вона є більш складною у засвоєнні. Машинний код, написаний на Scala, складніший для читання й сприйняття.

## П'ЯТЬ ПОКОЛІНЬ МОВ ПРОГРАМУВАННЯ

П'ять поколінь мов програмування – це історична класифікація, яка заснована на основних характеристиках і особливостях мов програмування.

### Перше покоління (1950-ті роки)

Мови першого покоління називають машинними мовами. Вони складаються з набору машинних кодів, які безпосередньо розуміються комп'ютером. Мови першого покоління дуже низькорівневі і складні у вивченні та використанні. Їх використовують для написання програм для найпростіших електронних обчислювальних машин.

*Приклади мов першого покоління:*

- Машинний код;
- Ассемблер.

### Друге покоління (1950–1960-ті роки)

Мови другого покоління називають асемблерними мовами. Вони більш високорівневі, ніж машинні, оскільки використовують мнемонічні символи для позначення машинних кодів. Ці мови більш прості у вивченні та використанні, ніж машинні, але все ще досить низькорівневі. Їх використовують для написання програм для більш складних електронних обчислювальних машин.

*Приклади мов другого покоління:*

- Ассемблер;
- FORTRAN;
- COBOL.

### Третє покоління (1960–1970-ті роки)

Мови третього покоління називають мовами високого рівня. Вони більш високорівневі, ніж асемблерні мови, оскільки використовують англоподібні ключові слова та синтаксис. Мови високого рівня є набагато простішими у вивченні та використанні, ніж машинні та асемблерні. Їх використовують для написання програм для широкого спектра застосувань.

*Приклади мов третього покоління*:

- BASIC;
- Pascal;
- C.

### Четверте покоління (1970–1990-ті роки)

Мови четвертого покоління називають об'єктно-орієнтованими мовами програмування. Вони додають до мов високого рівня підтримку об'єктно-орієнтованого програмування, яке є потужною технікою для розробки програмного забезпечення. Об'єктно-орієнтовані мови програмування більш гнучкі і ефективні, ніж традиційні мови високого рівня. Їх використовують для написання програм для широкого спектра застосувань.

*Приклади мов четвертого покоління:*

- Smalltalk;
- C++;
- Java.

### П'яте покоління (1990-ті – до сьогодні)

Мови п'ятого покоління називають інтелектуальними мовами програмування. Вони використовують технології штучного інтелекту для автоматизації завдань програмування. Інтелектуальні мови програмування ще перебувають на ранніх стадіях розробки, але мають потенціал значно спростити процес програмування.

*Приклади мов п'ятого покоління:*

- Prolog
- LISP
- Haskell.

Покоління мов програмування є важливим етапом у розвитку програмування. Кожне нове покоління мов програмування додавало нові можливості та вдосконалення, що робило програмування простішим, ефективнішим і потужнішим.

Мови першого покоління зробили можливим створення перших електронних обчислювальних машин. Мови другого покоління дозволили розробити більш складні програми для цих машин. Мови третього покоління зробили програмування доступнішим для широкого кола людей. Мови четвертого покоління додали підтримку об'єктно-орієнтованого програмування, що стало стандартом для розробки програмного забезпечення. Мови п'ятого покоління мають потенціал революціонізувати процес програмування.

## МОВИ ПРОГРАМУВАННЯ КОМП'ЮТЕРНИХ ІГОР

Браузерні ігри запускаються тільки через браузер, мобільні – зі смартфонів або планшетів, комп'ютерні – з ноутбуків і персонального комп'ютера (ПК), а консольні – через Sony PlayStation і Xbox та ін.

Для кожного типу використовують різні мови програмування розробки ігор: браузерні – на JavaScript, HTML5 або PHP; комп'ютерні – на C++, C# та Java: мобільні – на C#, JavaScript, C++ чи Java.

Незважаючи на те що C++ вважається відносно складною мовою для розробників відеоігор, його широко використовують великі розробники. На цій мові програмування написані двигуни Cry Engine (шутер Crysis) та Havok (рпг–гра Dark Souls).

Популярною у розробників відеоігор є Lua – вбудована мова. Складно знайти гру, де вона б не використовувалася для написання логіки та сценарію гри. Наприклад, цю мову використовували для створення режимів користувача в Dota 2.

## КОНТРОЛЬНІ ЗАПИТАННЯ ТА ЗАВДАННЯ

***1. Що являє собою мова програмування?***

***2. Чим мови програмування високого рівня відрізняються від мови програмування низького рівня?***

***3. Для чого використовують транслятори?***

***4. Що визначає синтаксис мови програмування?***

***5. Як можна описати роботу транслятора?***

***6. Яка мова програмування низького рівня залишається затребуваною до сьогодні?***

***7. Вкажіть недоліки мови програмування високого рівня.***

***8. Сформулюйте особливості мови програмування залежно від покоління, до якого вона віднесена.***

***9. Що відомо про мови програмування п'ятого покоління?***

***10. Назвіть мови програмування, які використовують розробники відеоігор.***

# ЛЕКЦІЯ 2. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПРИКЛАДНИХ ЗАДАЧ У ФІЗИЧНІЙ КУЛЬТУРІ І СПОРТІ. СУЧАСНІ КОМП'ЮТЕРНІ СИСТЕМИ В КІБЕРСПОРТІ

***1. Види прикладного програмного забезпечення.***
***2. Стан розробленості прикладного програмного забезпечення системи фізичної культури і спорту.***
***3. Огляд прикладних комп'ютерних програм, розроблених фахівцями Національного університету фізичного виховання і спорту України.***
***4. Прикладне програмне забезпечення сфери кіберспорту.***

## ВИДИ ПРИКЛАДНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Прикладне програмне забезпечення (ППЗ) – це сукупність комп'ютерних програм (додатків), спрямованих на вирішення прикладних задач (завдань користувача) (рис. 1.19).

Результат систематизації прикладного програмного забезпечення представлено в таблиці (табл. 1.4).

На відміну від програм загального призначення, програми, які використовуються вузьким колом фахівців, називаються програмами спеціального призначення.

**Прикладне програмне забезпечення спеціального призначення** – це сукупність програм на вирішення конкретних завдань різних предметних галузей (табл. 1.5).

Крім того, на практиці зустрічаються професійно орієнтовані задачі, які не можна вирішити за допомогою існуючого прикладного програмного забезпечення. Результати виходять у формі, яка не задовольняє кінцевого користувача. В цьому випадку за допомогою систем програмування або алгоритмічних

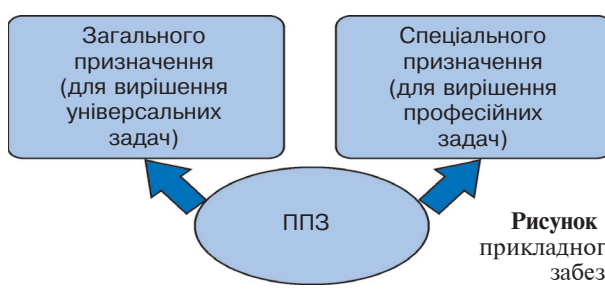



**Рисунок 1.19** – Види прикладного програмного забезпечення

**Таблиця 1.4** – Систематизація видів прикладного програмного забезпечення

| № з/п | Вид прикладного програмного забезпечення | Призначення | Приклади програм |
|---|---|---|---|
| 1 | Текстові редактори | Для створення і редагування тексту без оформлення | Notepad або Блокнот (входить в ОС MS Windows),TextPad |
| 2 | Текстові процесори | Для створення і редагування тексту з оформленням | MS Word, WordPad (входить в ОС MS Windows) |
| 3 | Електронні таблиці | Для обробки даних у табличній формі | MS Excel |
| 4 | Графічні редактори | Для створення і редагування зображень | |
| 4.1 | Растрові | Для роботи з растровими зображеннями | MS Paint (входить в ОС MS Windows), Adobe Photoshop |
| 4.2 | Векторні | Для роботи з векторними зображеннями | CorelDRAW, Adobe Illustrator |
| 5 | «Переглядачі» | Для перегляду файлів універсальних форматів | |
| 5.1 | «Переглядачі зображень» | Для перегляду зображень | CDSee, FastStone Image Viewer, FastPictureViewer |
| 5.2 | «Переглядачі» HTML-сторінок (браузери, веб-оглядачі) | Для перегляду сторінок веб-сайтів | MS Internet Explorer, Mozila Firefox, Google Chrome, Opera |
| 5.3 | «Переглядачі» медіа контенту (Медіаплеєри, медіапрогравачі) | Для відтворення медіа контенту | |
| | • аудіоплеєри | Для відтворення аудіофайлів | AIMP, Foobar2000, Spider player,MusicBee, Media Monkey |
| | • мультімедіа-центри | Для відтворення відео- та аудіофайлів | Windows Media Player (WMP, входит в ОС MS Windows), QuickTime Player (входит в ОС Mac OS X), Winamp, VLC media player, Media Player Classic |
| 5.4 | «Переглядачі» flash-контенту (Flash-плеєри) | Для відтворення відео і аудіо файлів на веб-сайтах, для ігор он-лайн | Adobe Flash Player |
| 5.5 | «Переглядач» pdf-файлів | Для перегляду і друку pdf-файлів | Adobe Reader |
| 6 | Системи управління базами даних | Для управління виробництвом і роботою з базами даних | MS Access, Paradox |
| 7 | Комп'ютерні ігри | Для розваги або навчання | |

*Продовження таблиці 1.4*

| № з/п | Вид прикладного програмного забезпечення | Призначення | Приклади програм |
|---|---|---|---|
| 8 | Перекладачі | Для перекладу окремих слів, текстів | ABBYY Lingvo, МультиЛекс, ПРОМТ |

**Таблиця 1.5** – Окремі види прикладного програмного забезпечення спеціального призначення

| № з/п | Окремі види прикладного програмного забезпечення спеціального призначення | Призначення | Приклади програм |
|---|---|---|---|
| 1 | Настільні видавничі системи | Для електронного верстання газет, журналів, книг, буклетів | QuarkXPress, Adobe InDesign, Adobe FrameMaker, Corel Ventura, MS Publisher |
| 2 | Електронні енциклопедії, підручники, словники, довідники | Для отримання знань в певній сфері | «Довідник майстра–будівельника», «Музичний словник» |
| 3 | Системи автоматизованого перекладу | Для перекладу професійних текстів | Trados, Deja Vu, Star Transit |
| 4 | Редактори тривимірної (3D) графіки | Для створення і редагування тривимірної графіки | Autodesk 3ds Max(раніше 3D Studio MAX), Autodesk Maya, Blender, Cinema 4D, ZBrush |
| 5 | Системи автоматизованого проєктування | Для розробки на комп'ютері креслень, схем, 3D-моделей, конструкторської та технологічної документації | Компас, AutoCAD, ZwCAD, nanoCAD |
| 6 | Геоінформаційні системи | Для створення, редагування та аналізу електронних географічних карт | MapInfo, CREDO_DAT, ArcGIS, Arcview, GeoServer, GRASS, gvSIG, Арго, Полігон, Панорама, ГИС Метео |

мов розробляються оригінальні програми, що враховують вимоги та умови вирішення задачі.

## СТАН РОЗРОБЛЕНОСТІ ПРИКЛАДНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ФІЗИЧНОЇ КУЛЬТУРИ І СПОРТУ

Серед напрямів використання прикладного програмного забезпечення в спорті можна виділити такі (рис. 1.20).

Прикладне програмне забезпечення широко використовується й у фізичному вихованні (рис. 1.21).

Розглянемо прикладне програмне забезпечення оздоровчого призначення (рис. 1.22).


**Рисунок 1.20** – Напрями використання прикладного програмного забезпечення в спорті


**Рисунок 1.21**– Напрями використання прикладного програмного забезпечення у фізичному вихованні

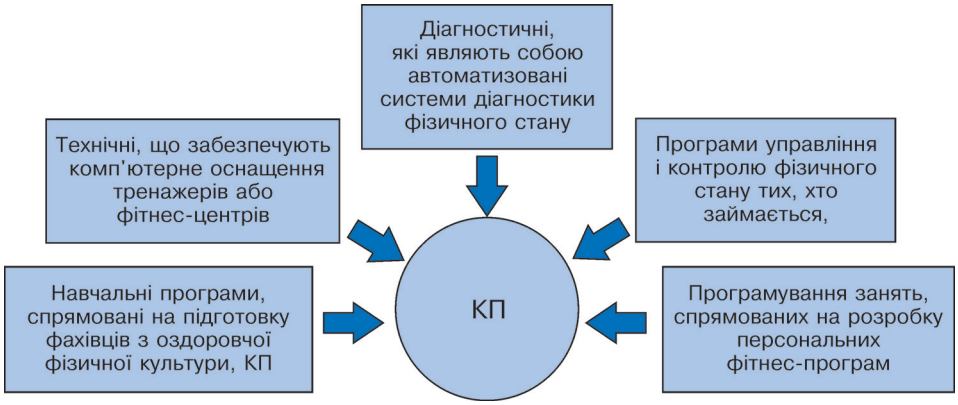



**Рисунок 1.22** – Комп'ютерні програми оздоровчого призначення

Окремі приклади прикладного програмного забезпечення оздоровчого спрямування наведено у таблиці 1.6.

Наведемо приклади прикладного програмного забезпечення сфери фізичної культури і спорту (рис. 1.23).

Програма **SwimBase** призначена для автоматизації суддівства змагань з плавання та збору і зберігання інформації про змагання спортсменів за великий період часу. Крім того, вона успішно застосовується у басейнах і спортивних школах. Там її використовують для ведення списків груп та друку перепусток, карток, абонементів тощо.

**Таблиця 1.6** – Окремі приклади прикладного програмного забезпечення оздоровчого спрямування

| № з/п | Окремі види прикладного програмного забезпечення оздоровчого спрямування | Призначення | Автор |
|---|---|---|---|
| 1 | «Fitness Centre» | Моделювання занять з урахуванням інтересів та рівня підготовленості учасників | О. С. Губарєва, 2001 |
| 2 | «Фітнес для жінок» | Відстеження динаміки змін показників фізичної підготовленості жінок (на основі баз даних) | Д. Ю. Луценко, 2003 |
| 3 | «Експрес-оцінка фізичного здоров'я школярів» | Моніторинг фізичного здоров'я школярів, індивідуалізація процесу фізичного виховання й оцінювання його оздоровчої ефективності | А. М. Соболев, 2006 |
| 4 | «Fitball training» | Виявлення динаміки рівня здоров'я, фізичного розвитку і фізичної підготовленості жінок | О. Ю. Лядська, 2010 |
| 5 | «Олімп» | Оцінювання і корекція харчування осіб, які займаються фітнесом | О. І. Циганенко, 2012 |
| 6 | «Фітнес-інструктор» | Складання програми індивідуальних тренувань на основі аналізу індивідуальних особливостей і показників здоров'я тих, хто займається | М. М. Гладишева, 2015 |
| 7 | «Фітнес-клас» | Визначення соматотипу жінок та відповідно до нього добирання раціональних параметрів фізичних навантажень для занять класичною аеробікою та степ-аеробікою | Н. В. Зінченко, 2016 |

Програмне забезпечення **SportsCode** призначене для просторово-часових видів спорту. Дозволяє здійснювати відеозапис і фіксацію події на спортивному майданчику, який зберігається для подальшого запиту, відтворення, перегляду й аналізу моделі руху та позиції як окремих гравців, так і їхніх груп (рис. 1.24).

Комп'ютерна програма **Home Workout** дозволяє користувачу підтримувати фізичну форму. Вона включає розминку і розтяжку та дозволяє автоматично записувати тренувальний процес, який потім можна проаналізувати й зробити відповідні висновки (рис. 1.25).

Огляд програмних продуктів моніторингового та інформаційного супроводу формування просторової організації тіла людини представлено на рисунку 1.26.

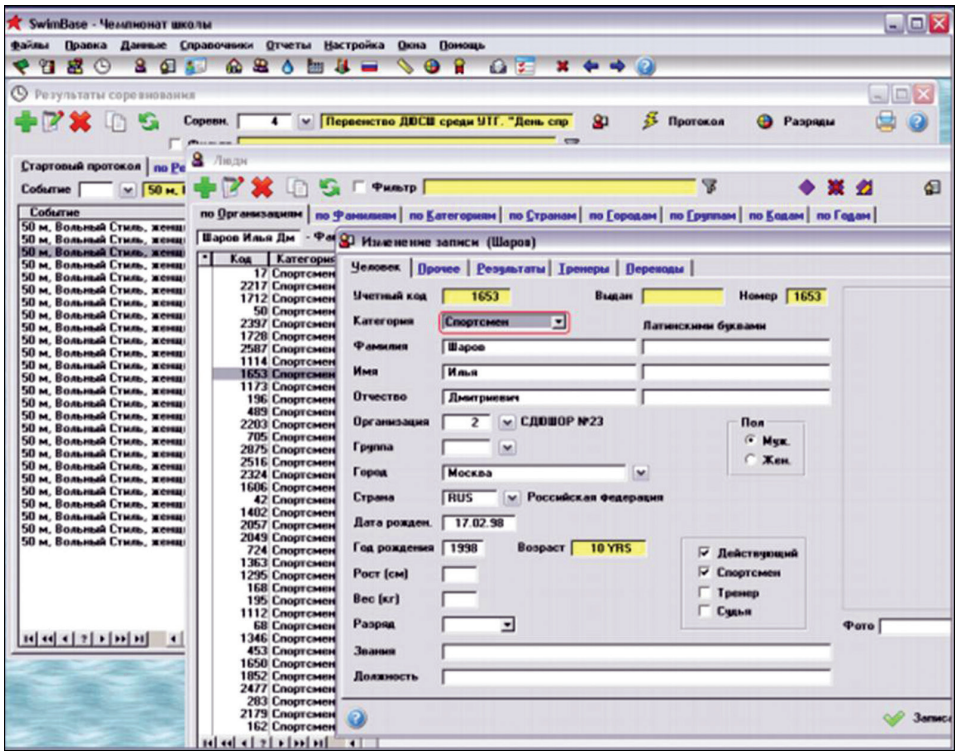



**Рисунок 1.23** – Інтерфейс програми SwimBase (розробник PSoft)

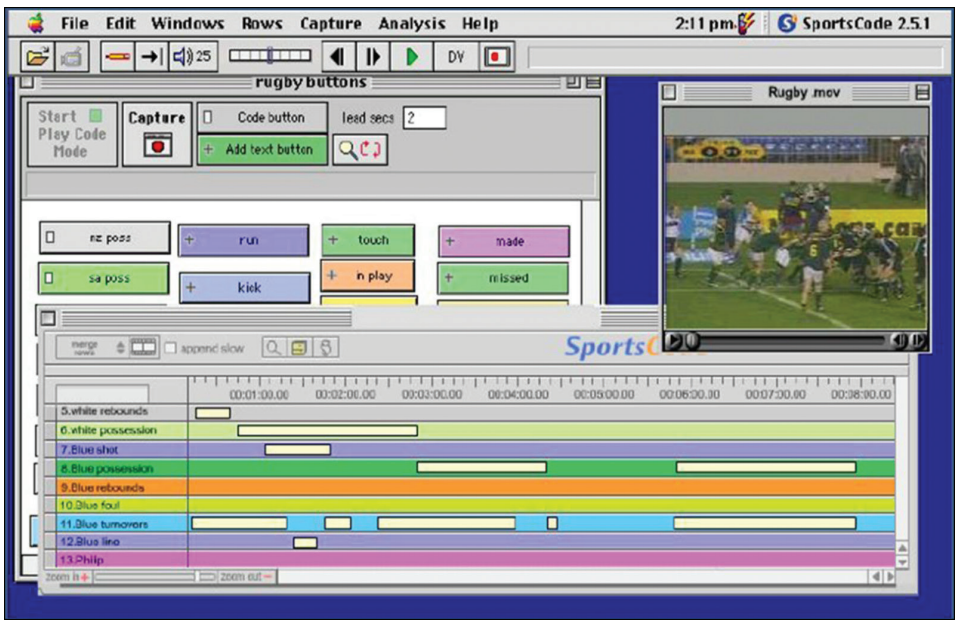



**Рисунок 1.24** – Знімок екрану SportsCode (SportsTec, 2001)


**Рисунок 1.25** – Додаток Home Workout для персонального комп'ютера


**Рисунок 1.26** – Програмні продукти моніторингового інформаційного супроводу

## ОГЛЯД ПРИКЛАДНИХ КОМП'ЮТЕРНИХ ПРОГРАМ, РОЗРОБЛЕНИХ ФАХІВЦЯМИ НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ФІЗИЧНОГО ВИХОВАННЯ І СПОРТУ УКРАЇНИ

Фахівці Національного університету фізичного виховання і спорту не залишилися осторонь тенденції до впровадження інформаційних технологій (ІТ) у сферу фізичного виховання і спорту. Зробимо огляд комп'ютерних програм, розроблених на базі університету.

З метою автоматизації обробки відеограм біогеометричного профілю постави відносно сагітальної й фронтальної площини запропоновано програму **TORSO** (рис. 1.27).

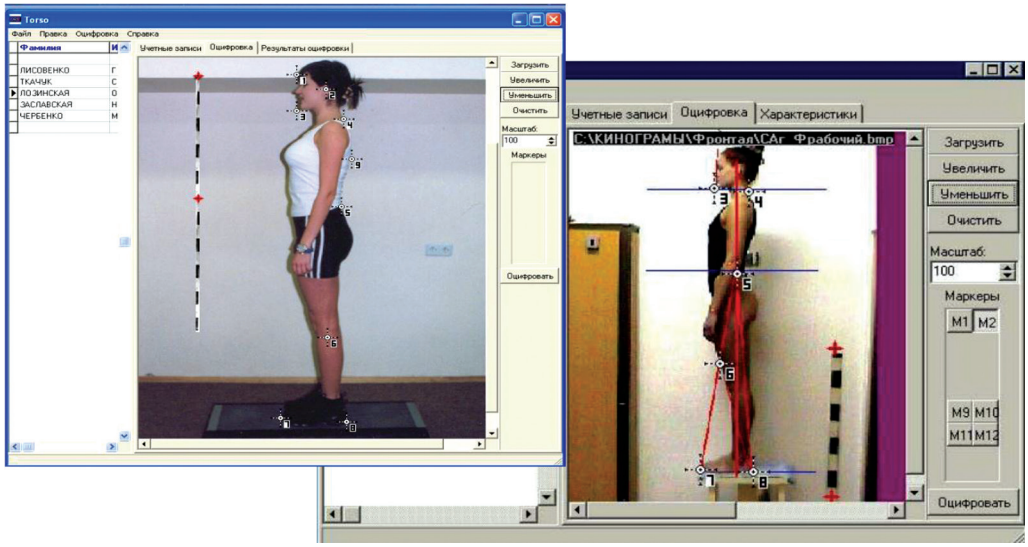



**Рисунок 1.27** – Знімки з екрана програми TORSO (за: В. О. Кашуба, 2003)

З метою автоматизації обробки плантограм людини та визначення морфобіомеханічних характеристик стопи розроблено комп'ютерну програму **FOOT–PRINT** (рис. 1.28).

Для аналізу скелетних компонентів стопи, які забезпечують її опорно–ресорну функцію, розроблено комп'ютерну програму **Big Foot** (рис. 1.29).

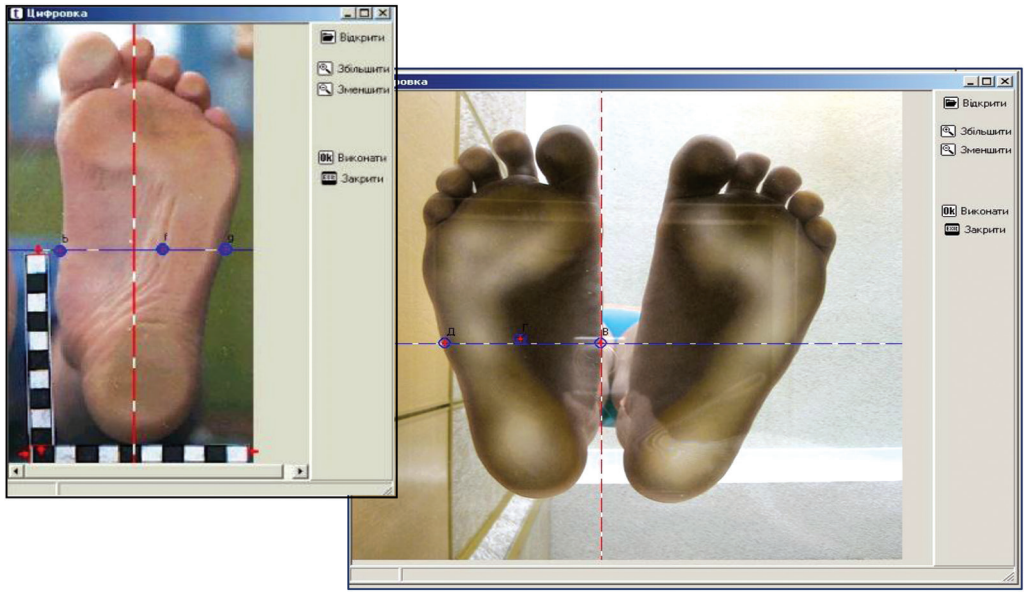



**Рисунок 1.28** – Знімки з екрана програми FOOT–PRINT (за: В. О. Кашуба, 2004)

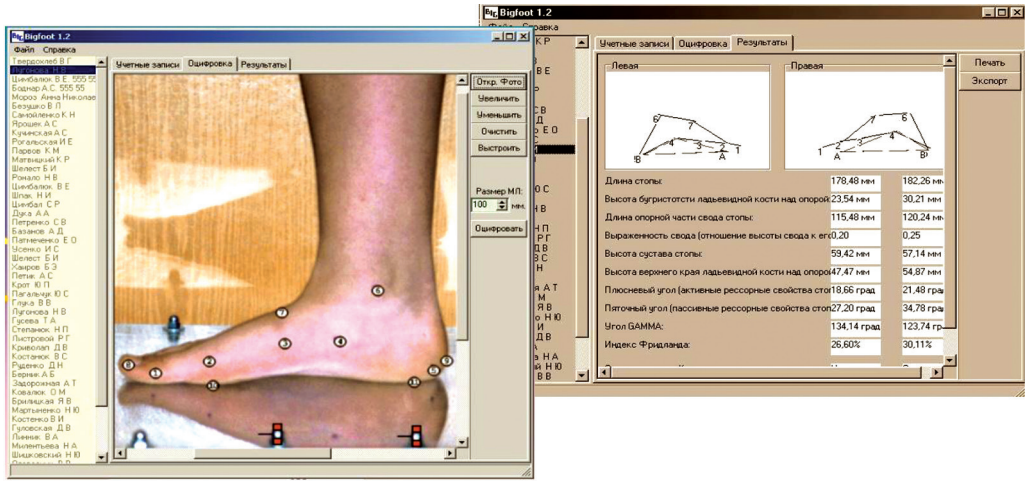



**Рисунок 1.29** – Знімки з екрана програми Big Foot (за: К. М. Сергієнко, 2002)

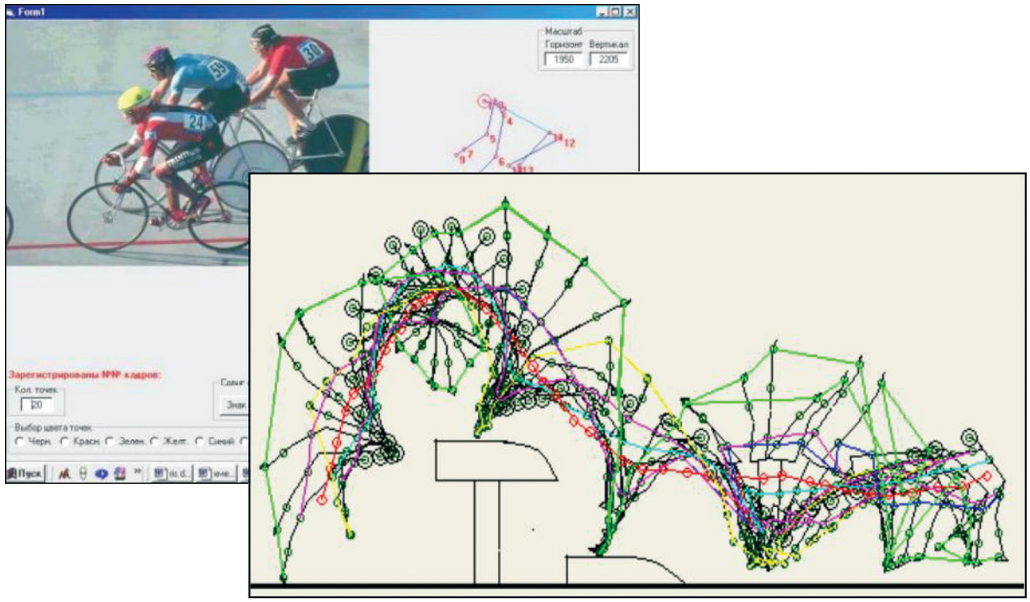



**Рисунок 1.30** – Знімки з екрана пакета прикладних програм БіоВідео (за: І. В. Хмельницька, 2002)

Отримати біомеханічні характеристики як окремих біоланок, так і всього тіла людини в кожному кадрі та в окремих фазах рухової дії дозволяє комп'ютерна програма **БіоВідео** (рис. 1.30).

Визначити еталонні обхватні розміри біоланок бодибілдерів високої кваліфікації відповідно до їхніх індивідуальних масо-зростових показників допомагає комп'ютерна програма **Атлет** (рис. 1.31).

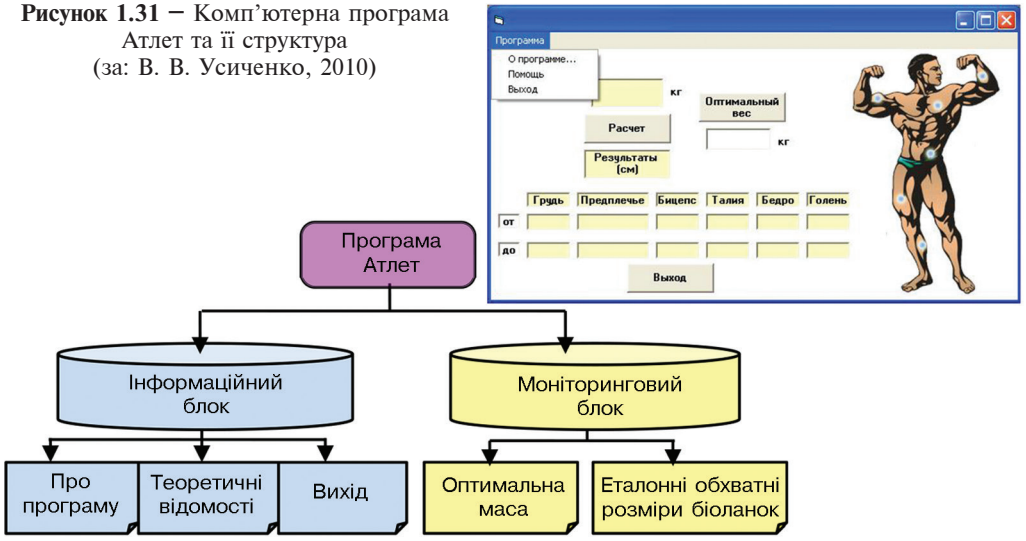



**Рисунок 1.31** – Комп'ютерна програма Атлет та її структура (за: В. В. Усиченко, 2010)

Зауважимо, що в основу регресійних моделей, на основі яких здійснюють розрахунки, покладено антропометричні показники усіх найбільш відомих бодибілдерів світового рівня, включаючи Арнольда Шварцнегера.

За допомогою бази даних **Календар тренувань** можна здійснювати збір і аналіз інформації про чоловіків, які відвідують тренажерний зал, засобів тренувань та обсягів виконаних навантажень (рис. 1.32).

Для залучення жінок першого зрілого віку до занять пілатесом шляхом розширення частки самостійності жінок у процесі тренування розроблено комп'ютерну фітнес-програму **PILATES** (рис. 1.33).

Зі структурою запропонованої комп'ютерної фітнес-програми PILATES можна ознайомитися за допомогою таблиці (табл. 1.7).

Для встановлення рівня стану біогеометричного профілю постави молодших школярів та розробки рекомендацій до профілактики пору-

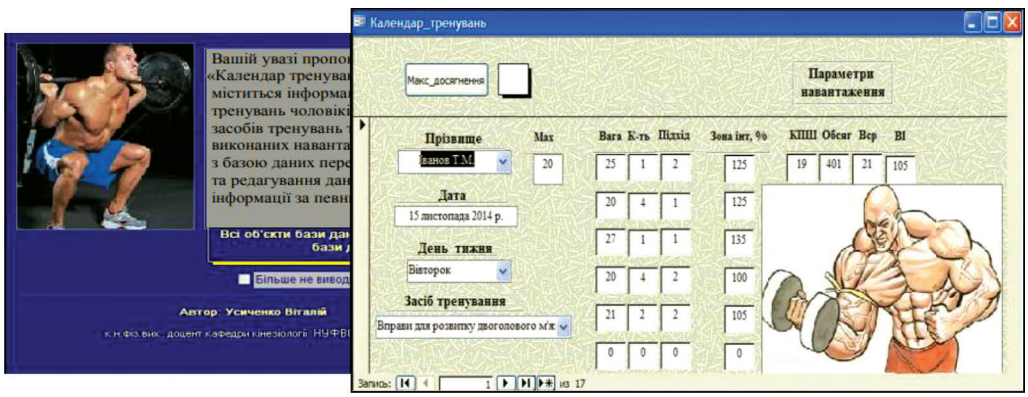



**Рисунок 1.32** – База даних Календар тренувань (за: В. В. Усиченко, 2015)

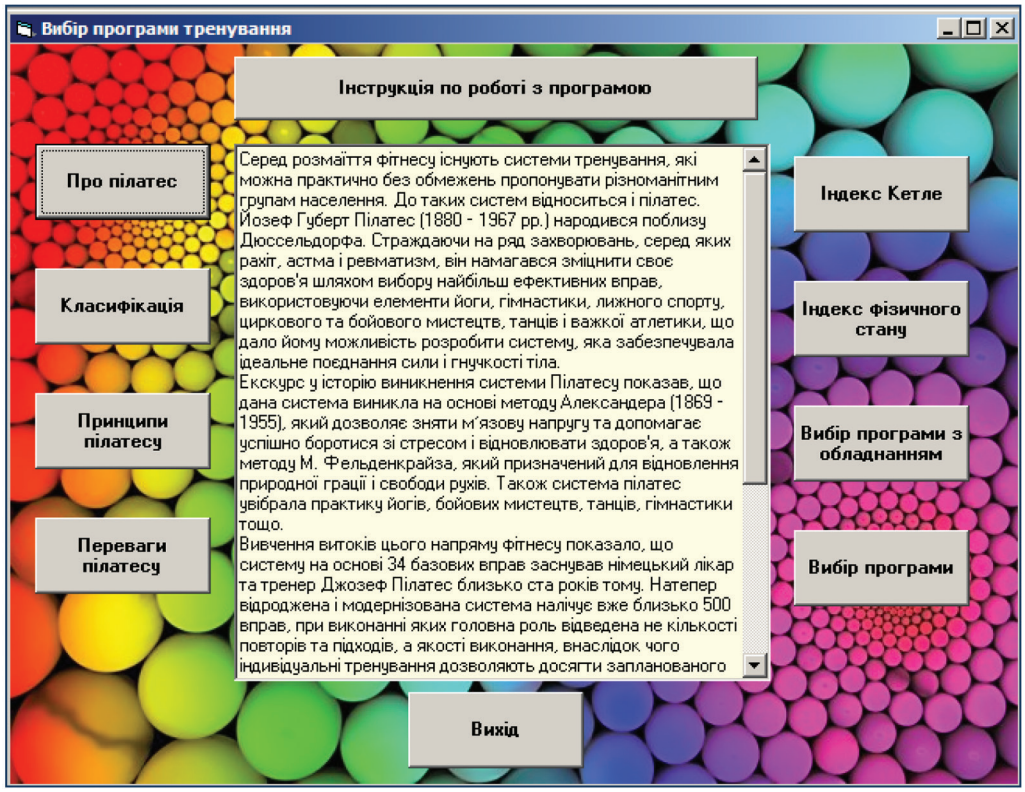



**Рисунок 1.33** – Головне вікно програми PILATES (за: Ю. Томіліна, 2016)

шень їхньої постави запропоновано інформаційно-аналітичну систему **Habitus** (рис. 1.34).

**Таблиця 1.7** – Структура комп'ютерної фітнес-програми PILATES

| Блок | Розділ програми | Засоби |
|---|---|---|
| Інформаційний | «Інструкція», «Система Пілатес», «Класифікація вправ», «Принципи пілатесу», «Переваги пілатесу» | Навчально-методичні та довідкові матеріали |
| Розрахунковий | «Індекс Кетле», «Індекс фізичного стану» | Розрахункові формули |
| Фізкультурно–оздоровчий | «Вибір програми з обладнанням», «Вибір програми без обладнання», «Вибір програми з урахуванням РФС» | Комплекси фізичних вправ для жінок з різним рівнем фізичного стану та залежно від рухових уподобань |

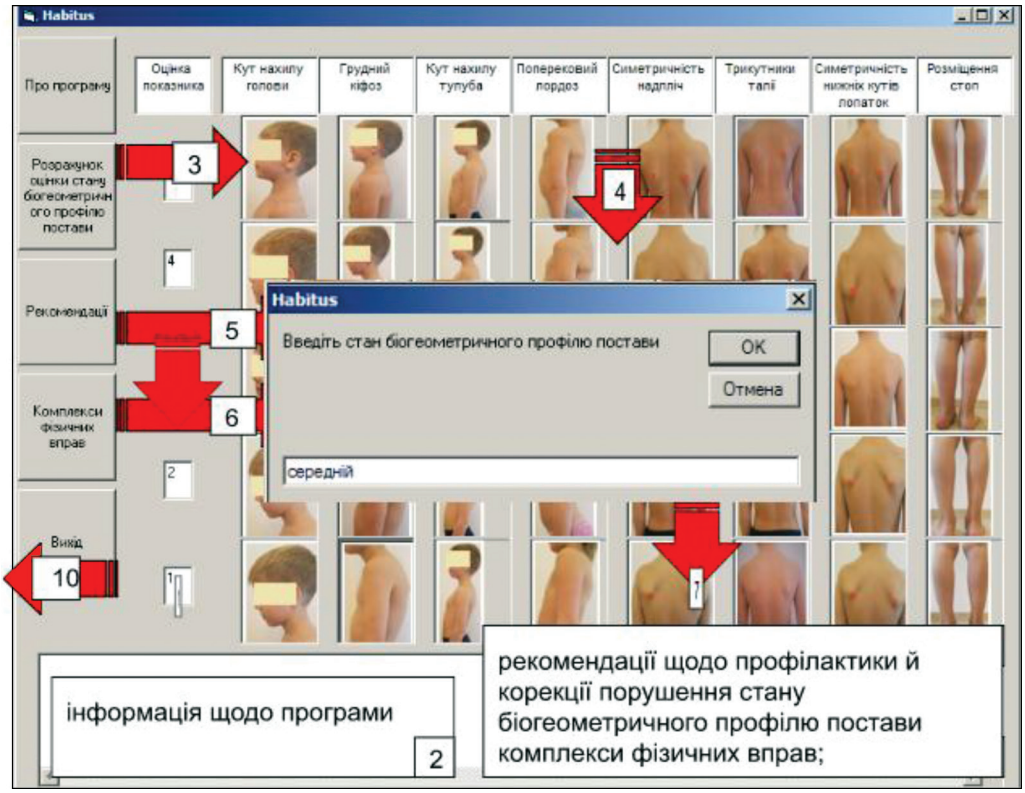



**Рисунок 1.34** – Головне вікно та етапи роботи з інформаційно–аналітичною системою Habitus (за: Н. Л. Носова, 2021)

## ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СФЕРИ КІБЕРСПОРТУ

Як і в інших випадках, прикладне програмне забезпечення, що використовується у сфері кіберспорту, поділяється на ППЗ загального і спеціального призначення (рис. 1.35).

Розглянемо приклади прикладного програмного забезпечення сфери кіберспорту. Так, ігрова платформа **Discord** – це прикладне програмне забезпечення для управління кіберспортивними дисциплінами. Вона надає організаторам

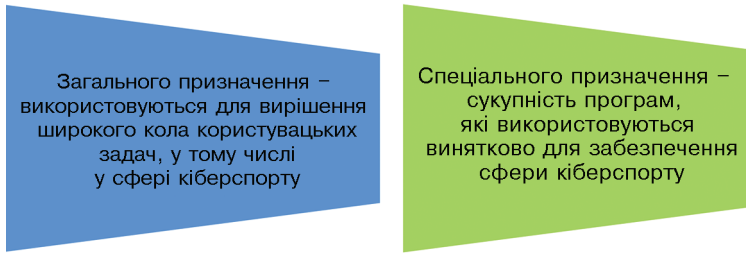



**Рисунок 1.35** – Види прикладного програмного забезпечення

спортивних заходів засоби для планування, просування та управління кіберспортивними подіями, турнірами, змаганнями, командами та гравцями (рис. 1.36).

Онлайн-платформа **Brackot** створена для управління турнірами та змаганнями, вона розробляє передові інструменти для організаторів і забезпечує організацію турнірів як для окремих гравців, так і для команд (рис. 1.37).

На сьогодні функціонує навчальна платформа «**Залучайте своїх учнів до кіберспорту**» (**Engage Your Students With Esports),** що пропонує доєднатися до спіль-

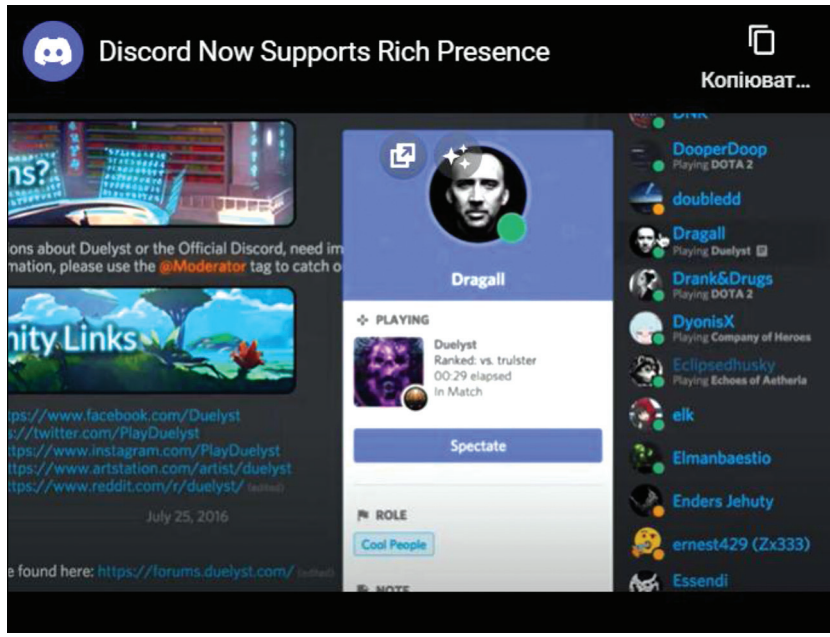



**Рисунок 1.36** – Вид прикладного програмного забезпечення Discord

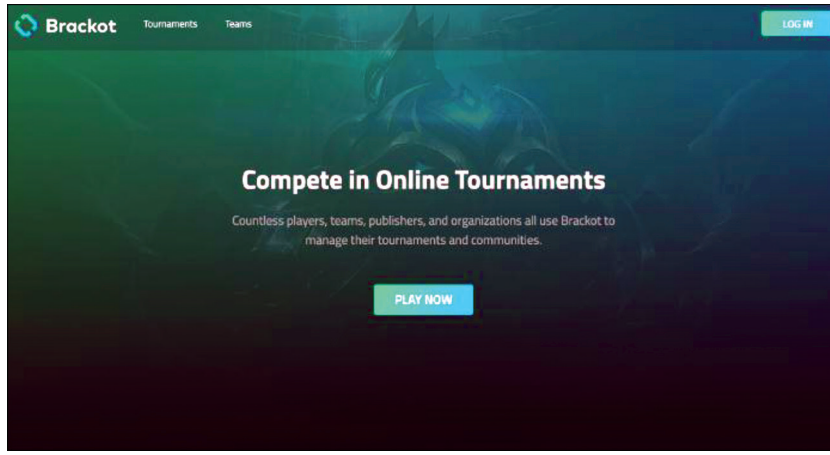



**Рисунок 1.37** – Онлайн-платформа Brackot (United States, 2019)


**Рисунок 1.38** – Навчальна платформа «Залучайте своїх учнів до кіберспорту» (Generation Esports, 2017)

ноти із понад 3 000 шкіл, беручи участь у кіберспортивних змаганнях, анонси майбутніх подій, актуальні статті тощо. Бібліотеку навчальних програм створено професійними тренерами та спортсменами з кіберспорту південно-корейської кіберспортивної організації Generation Esports, які популяризують кіберспорт та здійснюють підготовку наступного покоління кіберспортсменів. Програми навчання охоплюють найпопулярніші ігри та всі рівні навичок (рис. 1.38).

У розділі *Навчання* можна дізнатися, як підготуватися до турніру, як працювати з командою та створювати підпрограми, тут подано інформацію про кіберспортивну етику та особливості психології кіберспорту, розглянуто питання про харчування та гігієну сну кіберспортсменів, а також загострено увагу на багатьох інших актуальних питаннях кіберспорту (рис. 1.39).

У цьому розділі представлено такі інструменти для навчання кіберспорту, як Aim Lab та Game Reviews.

**Aim Lab** – провідний інструмент зорово-моторної координації, що використовується як професійними спортсменами, так і аматорами для покращення прицілювання та механіки роботи з мишею.

Він включає:

- цільові програми навчання для кожного рівня кваліфікації;
- вправи для розминки;
- статистику для аналізу продуктивності у часі.

**Game Reviews** – огляди ігор, спрямовані на вдосконалення шляхом аналізу ігрових матчів, котрі можуть переглядатися інструктором, учнями або професій-ним тренером. Game Reviews дозволяє:

- виділити модельні ігри та навички, які потрібно покращити, коментуючи відео;

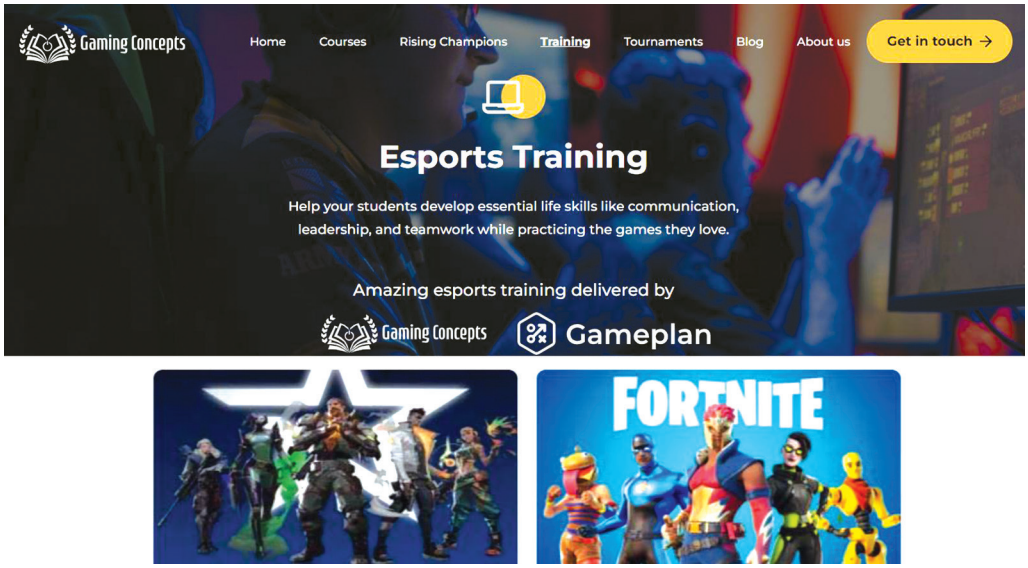








**Рисунок 1.39** – Інструменти для навчання кіберспорту

• підкреслити свою точку зору, малюючи на екрані;
• здійснювати спільну роботу в режимі реального часу.

Науково-педагогічні працівники Національного університету фізичного виховання і спорту України разом зі студентами, які навчаються за освітньо-професійною програмою «Кіберспорт (esports)» другого магістерського рівня вищої освіти спеціальності 017 Фізична культура і спорт, також долучилися до створення комп'ютерних програм для сфери кіберспорту.

Із метою розширення теоретичної бази зацікавлених осіб із питань еволюції кіберспорту розроблено інформаційну комп'ютерну програму «Еволюція кіберспорту» (рис. 1.40).

За допомогою програми можна ознайомитися з відеоіграми, які великою мірою вплинули на еволюцію кіберспорту, й інформацією, яка представляє інтерес для розвитку кіберспортивної науки.

Програма містить 14 керуючих кнопок та вікно для виведення інформації.

За допомогою кнопки «Інструкція про роботу з програмою» користувач може ознайомитися з інформацією про розробника програми, про її структуру й спрямування, а також дізнатися, як власне працювати з програмою.

Кнопка «Історія розвитку кіберспорту» запускає у вікні для виведення інформації систематизовані й стислі відомості про еволюцію кіберспорту – від ретро-ігор до кіберспортивних дисциплін.

За допомогою керуючих елементів, зокрема кнопок Скасувати/Вийти користувач може або очистити вікно для виведення інформації, або вийти з програми.

Решта керуючих кнопок містять назву тієї чи іншої ретро-гри або сучасної кіберспортивної дисципліни й, натискаючи її, користувач отримує відомості про специфічні особливості гри, сюжет, основних героїв, їхні навички

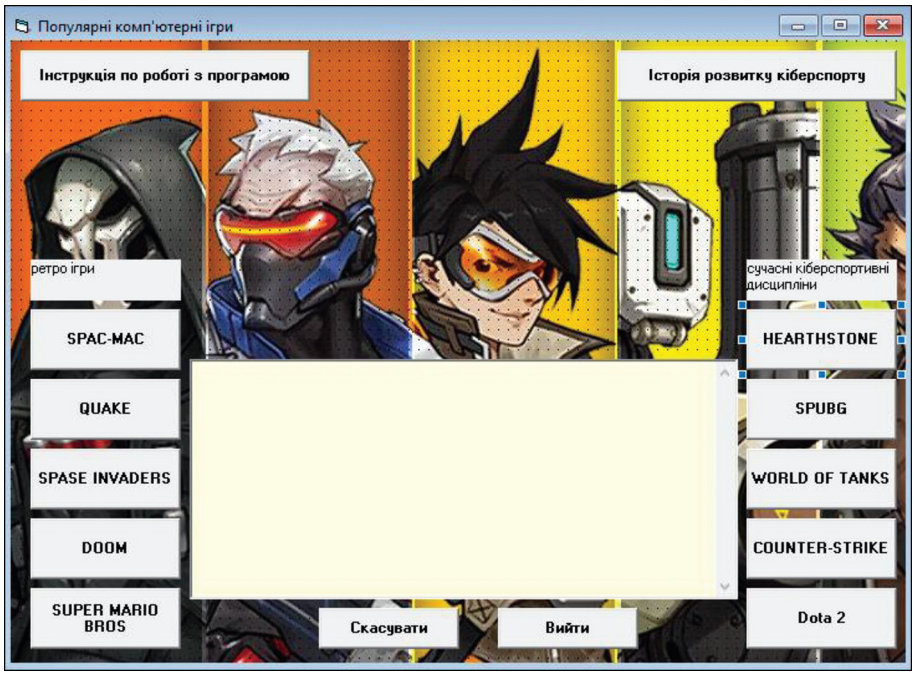



**Рисунок 1.40** – Головна панель інформаційної комп'ютерної програми «Еволюція кіберспорту» (за: Н. Г. Бишевець, Л. Орловська, 2021)

й здібності, а також може дізнатися про налаштування гри, правила змагань, рівень поширення та статистику популярності й максимальний розмір призового фонду.

Структура запропонованої програми містить елементи, представлені на рисунку (рис. 1.41).

Для ознайомлення з основними факторами, що вплинули на розвиток кіберіндустрії, й окреслення перспективи впровадження технологічних інновацій, зокрема смарт-технологій, в тренувальний процес кіберспортсменів, запропоновано інформаційно-методичну систему *Кіберспорт:* історія, становлення та перспективи розвитку. Натискання клавішею миші на кожну з керуючих кнопок дозволяє задати виконання тієї чи іншої дії. Головне вікно програми з визначеними елементами керування представлено на рисунку (рис. 1.42).

Кнопки «Розвиток кіберіндустрії», «Смарт технології» та «Методика» – це керуючі кнопки, за допомогою яких користувач може ознайомитися зі схематичним представленням відповідних процесів. Зауважимо, що у системі передбачено таку дію: виклик підлеглих форм супроводжується приховуванням головного вікна.

Тоді на екрані комп'ютера користувач бачить лише підлеглу форму й під час ознайомлення з інформацією, яку вона містить, його увага не відволікається на інші елементи інформаційно-методичної системи (рис. 1.43).

Взаємодія користувача з кнопками «Смарт-годинники, смарт-браслети», «Смарт-ваги, смарт-виделка», «Смарт-екіпіровка», «Портативний апарат Chek»


**Рисунок 1.41** – Структура програми «Еволюція кіберспорту»

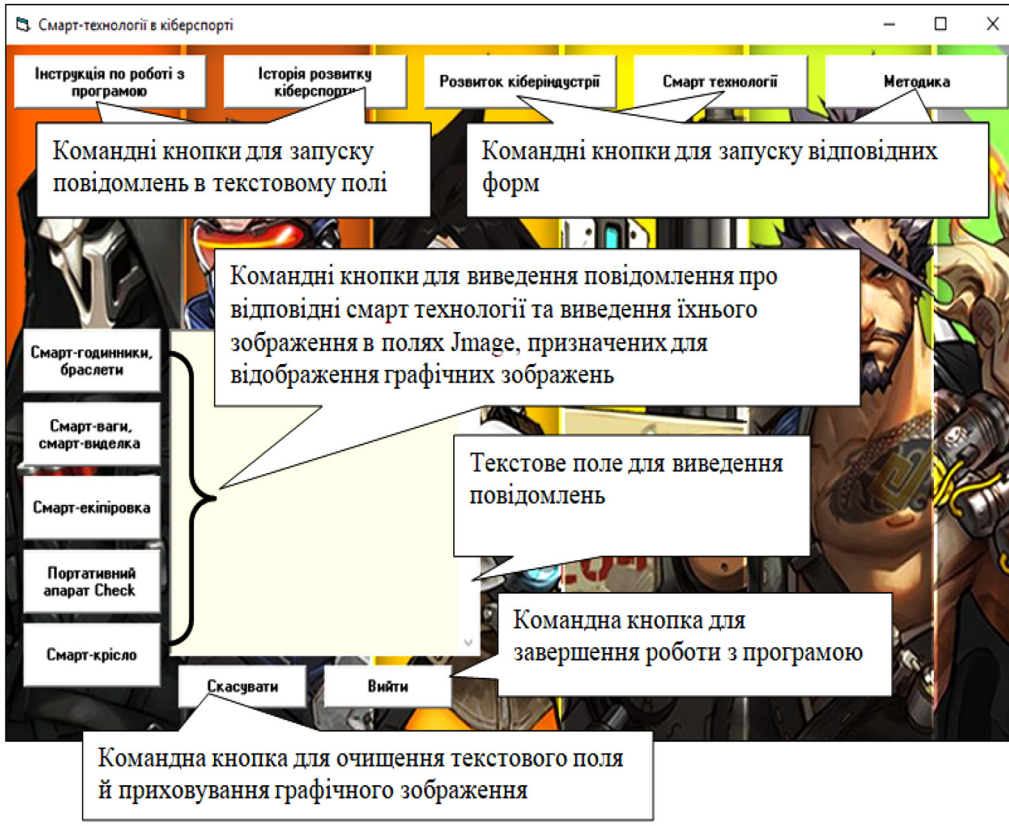



**Рисунок 1.42** – Вікно інформаційно-методичної системи Кіберспорт: історія, становлення та перспективи розвитку (за: Н. Г. Бишевець, І. Лендел, 2022)

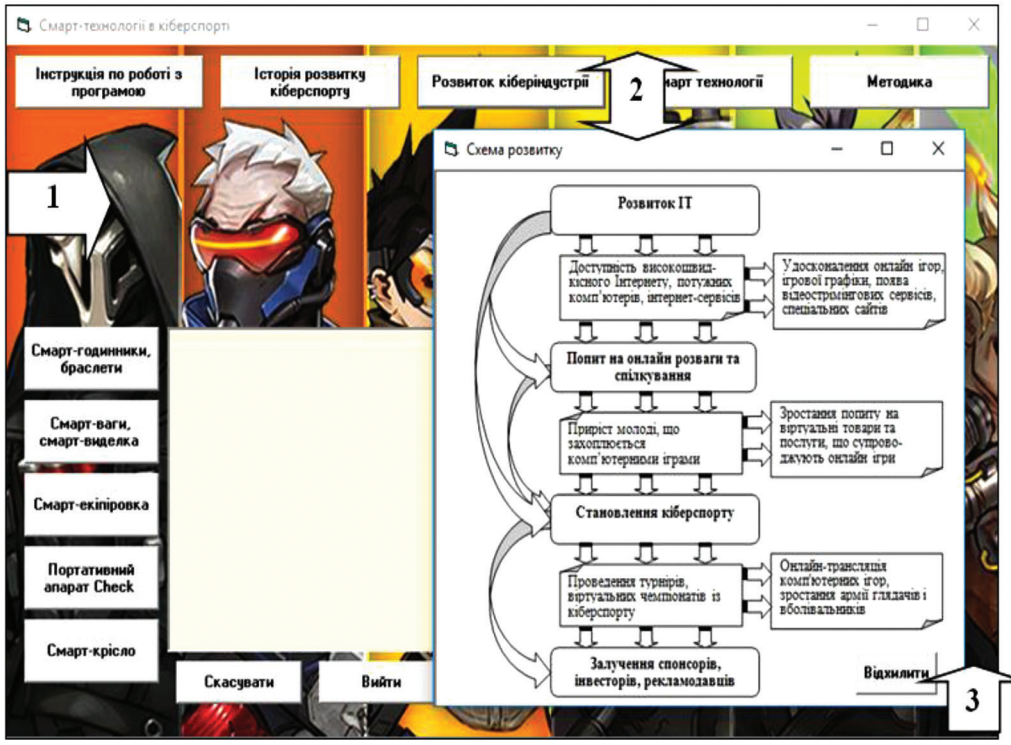



**Рисунок 1.43** – Робота з інформаційно-методичною системою Кіберспорт: історія, становлення та перспективи розвитку

та «Смарт-крісло» викликає повідомлення, які містять систематизовану інформацію про ту чи іншу технологію, а також їх візуальне представлення, приховане до натиснення кнопки.

З нашої точки зору, запропоновані комп'ютерні програми сприятимуть удосконаленню освітнього процесу під час підготовки майбутніх фахівців із кіберспорту, а також тренувального процесу кіберспортсменів та дозволять розширити теоретичні знання з історії кіберспорту, ознайомити зацікавлених осіб з різноманіттям смарт-технологій, які мають перспективи в практиці кіберспорту.

## ЗАВДАННЯ ДЛЯ ПРАКТИЧНИХ РОБІТ

**Практична робота № 1.** Розробити презентацію / доповідь / тези на тему «Перспективи застосування інформаційних технологій у сфері кіберспорту».

## КОНТРОЛЬНІ ЗАПИТАННЯ ТА ЗАВДАННЯ

***1. Які види прикладного програмного забезпечення вам відомі?***

***2. Чим вирізняється прикладне програмне забезпечення загального і спеціального призначення?***

***3. Наведіть приклади прикладного програмного забезпечення загального призначення.***

***4. Наведіть приклади прикладного програмного забезпечення спеціального призначення.***

***5. Яким чином використовується прикладне програмне забезпечення у спорті?***

***6. Яким чином використовується прикладне програмне забезпечення у фізичній культурі?***

***7. Як класифікують комп'ютерні програми оздоровчого спрямування?***

***8. Наведіть приклади прикладного програмного забезпечення оздоровчого спрямування.***

***9. Яким чином використовується прикладне програмне забезпечення у сфері кіберспорту?***

***10. Наведіть приклади прикладного програмного забезпечення сфери кіберспорту.***

ТЕМА 2

# ЗАСТОСУВАННЯ МЕТОДІВ СТАТИСТИЧНОГО АНАЛІЗУ ДАНИХ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧ КІБЕРСПОРТУ

## ЛЕКЦІЯ 3. ЗАСТОСУВАННЯ МЕТОДІВ СТАТИСТИЧНОГО АНАЛІЗУ ДАНИХ ДЛЯ РОЗВ'ЯЗАННЯ ПРИКЛАДНИХ ЗАДАЧ КІБЕРСПОРТУ

***1. Вимірювання. Види шкал.***
***2. Основні поняття аналізу даних.***
***3. Застосування методів статистики у розв'язанні складних задач кіберспорту.***
***4. Нормальний закон розподілу.***
***5. Виявлення тенденцій у розвитку кіберспорту.***

### ВИМІРЮВАННЯ. ВИДИ ШКАЛ

Бурхливий розвиток кіберспорту супроводжується появою багатьох проблем: необхідність його правового врегулювання, розробка освітніх програм для підготовки суддівського і тренерського складу даної спортивної дисципліни, визначення переліку препаратів, заборонених для кіберспортсменів, уточнення понятійного апарату для однозначного трактування явищ і процесів, пов'язаних із кіберспортивною індустрією. Відбувається формування основних понять й дефініцій кіберпортивної науки, створюється її понятійно-категоріальний апарат.

Розвиток кіберспортивної науки ознаменував початок ґрунтовних наукових досліджень у сфері кіберспорту. На тлі накопичення кіберспортивних даних виникла необхідність їх обробки.

Науковою основою для отримання надійних результатів експериментальної діяльності є математико-статистична обробка емпіричних даних, а застосування інформаційних технологій у процесі встановлення закономірностей і тенденцій у сфері кіберспорту відкриває значні перспективи для застосування математично-статистичних методів широкому колу дослідників (рис. 2.1).

**Вимірювання** – процес присвоєння чисел характеристикам об'єктів, що вивчаються, згідно з певним правилом.

**Номінальна шкала** (номінативна, найменувань) – це шкала, яка містить тільки категорії. Вимірювання в цій шкалі відбувається шляхом присвоєння досліджуваному об'єкту чи явищу визначеного позначення чи символу. Наприклад, рівні технічної підготовленості: високий, середній, низький. Включає дихотомічну, що містить лише дві категорії (наприклад, стать). Дані, виміряні у номінальній шкалі, не можна впорядкувати.

**Порядкова шкала** (ординарна, рангова) передбачає ранжування (упорядкування) досліджуваних об'єктів чи явищ. Для позначення відносної позиції об'єктам присвоюються числа. Проте мова не йде про величину відмінностей між ними. Як приклад даних, виміряних у порядковій шкалі, можна вказати місце кіберспортсмена у рейтингу.

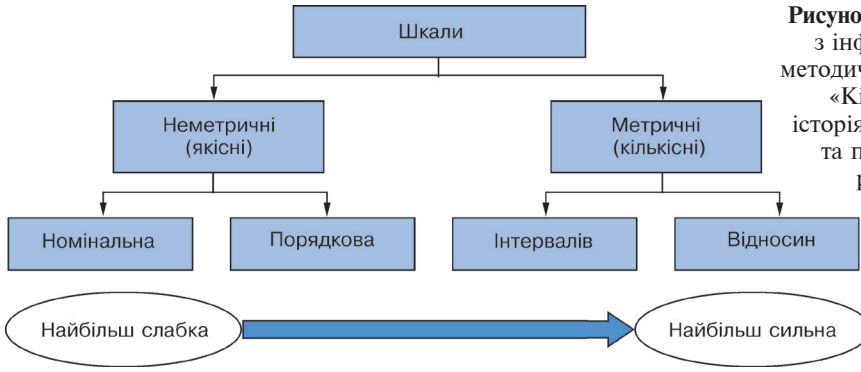



**Рисунок 2.1** – Робота з інформаційно-методичною системою «Кіберспорт»: історія, становлення та перспективи розвитку

**Інтервальну шкалу** розглядають як розширення порядкової шкали. У цій шкалі рівні впорядковані, а інтервали між ними рівні. Відмінності між значеннями можна обчислити, проте їх відношення не мають змісту. Прикладом вимірювання в інтервальній шкалі є IQ кіберспортсменів. Найбільш поширеним прикладом інтервальної шкали є шкала Лайкерта, яка побудована за 5-бальною системою.

**Відносна шкала** (шкала відносин) характеризується найбільш високим рівнем вимірювання змінних. Вона має всі властивості номінальної, порядкової та інтервальної шкал і, крім того, має точку початку відліку, «природний нуль». Наприклад, кількість вболівальників, які переглядали турнір, сума призових, швидкість реакції кіберспортсменів.

Номінальна і порядкова шкали називаються **неметричними**, а інтервальна та відносна – **метричними**.

## ОСНОВНІ ПОНЯТТЯ АНАЛІЗУ ДАНИХ

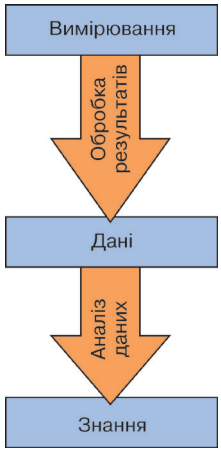



**Рисунок 2.2** – Етапи обробки результатів вимірювання

Для отримання знань результати вимірювання потребують обробки та статистичного аналізу (рис. 2.2).

**Дані** – необроблений матеріал, що використовують для формування інформації на їхній основі. Наприклад, дані – показники швидкості реакції кіберспортсменів, внесені у таблицю MS Excel.

Для оцінювання середньої швидкості реакції кіберспортсменів, встановлення закономірностей і тенденцій, прогнозування розвитку швидкості необхідна обробка даних, тобто аналіз даних або статистичний аналіз.

Аналіз даних наукових досліджень у кіберспорті – процес виявлення *невідомих, практично корисних і доступних для інтерпретації знань*, необхідних для прийняття рішень, зокрема для побудови тренувальних програм, прогнозування результатів турніру, визначення тенденцій розвитку кіберспорту. Це відносно новий напрям у науці прикладного спрямування, який не містить кінцевого набору базових фактів, з якого слідує, як розв'язувати задачі.

Як зазначалося, результатом успішного аналізу даних є отримання нових і корисних знань.

Якщо щось виглядає як комп'ютер, працює як комп'ютер, то, ймовірно, це і є комп'ютер – це *тривіальна інформація*. Якщо ймовірність того, що кіберспортсмен посяде перше місце у рейтингу рівна 0,8, то ймовірність, що він не посяде перше місце становить 0,2 – *відома інформація*. Якщо тренування у команді кіберспортсменів замінити на тренування з використанням програмного тренажера, то показники його технічної підготовленості не відрізнятимуться – *корисна інформація*. На рисунку 2.3 представлено відмінності між результатами аналізу даних.

## ЗАСТОСУВАННЯ МЕТОДІВ СТАТИСТИКИ У РОЗВ'ЯЗАННІ СКЛАДНИХ ЗАДАЧ КІБЕРСПОРТУ

Розглянемо типові задачі статистичного аналізу у кіберспорті (САК) (рис. 2.4). Коротко зупинимося на термінах, які застосовують у ході аналізу даних.

**Генеральна сукупність** – сукупність усіх об'єктів, які досліджуються, наприклад, усі кіберспортсмени, що потрапили до ТОП-50 з кібердисципліни Dota2.

**Вибірка** – частина генеральної сукупності, наприклад, усі кіберспортсмени, що потрапили до ТОП-50 з кібердисципліни Dota–2 за 2024 р.

**Репрезентативна вибірка** – вибірка, яка відображає характеристики генеральної сукупності.

**Параметри** – числові характеристики генеральної сукупності.

**Статистики** – числові характеристики вибіркової сукупності.

Аналізу даних передує висування гіпотези (нульова гіпотеза), яка підтверджується або відхиляється у ході статистичного аналізу.

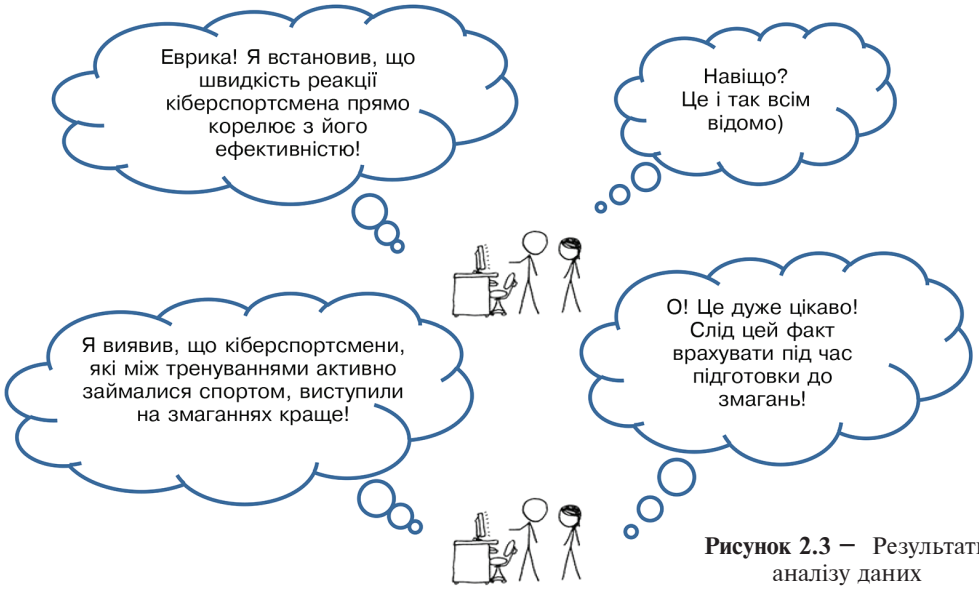



**Рисунок 2.3 –** Результати аналізу даних


**Рисунок 2.4** – Статистичний аналіз у кіберспорті

**Гіпотеза** – вислів, що містить припущення, тобто частково обґрунтована закономірність знань, яка служить для зв'язку між емпіричними даними або пояснює факти.

*Приклад гіпотези 1.* Рівень спортивних досягнень кіберспортсменів залежить від стресостійкості гравця.

*Приклад гіпотези 2.* Застосування тренувальної програми, заснованої на принципах оздоровчого тренування, дозволить підвищити ефективність кіберспортсменів і продовжити їхню спортивну кар'єру.

Вибір критерію для аналізу даних залежить від того, в якій шкалі отримано досліджувані показники.

**Кількісними показниками** називають показники, які вимірюються у метричній шкалі. Етапи аналізу кількісних показників представлено на рисунку 2.5.

**Якісні показники** вимірюють у номінальній або порядковій шкалі. Етапи аналізу якісних показників представлено на рисунку 2.6.

Перевірку статистичної гіпотези проводять для **прийнятого рівня значущості α** – граничне значення, що приймається до початку аналізу даних (береться рівним 0,1; 0,05; 0,01 тощо).

Прийнятий рівень значущості (p-рівень, p-значення) $\alpha = 0{,}05$ означає, що висунута нульова статистична гіпотеза може бути прийнята з довірчою ймовірністю $p = 0{,}95$.

Якщо p-значення, що спостерігається, менше альфа, то результат є статистично значущим ($p < 0{,}05$).

Запис «$p < 0{,}05$» означає, що при перевірці статистичної гіпотези є 5 % ймовірності відхилити правильну гіпотезу (зробити похибку першого роду) та $1–0{,}05 = 0{,}95 \cdot 100 = 95$ % – ймовірність прийняти правильне рішення.

У дослідженнях із фізичної культури і спорту найбільш поширеним є p-рівень, рівний 0,05.


**Рисунок 2.5** – Етапи аналізу кількісних даних у кіберспорті

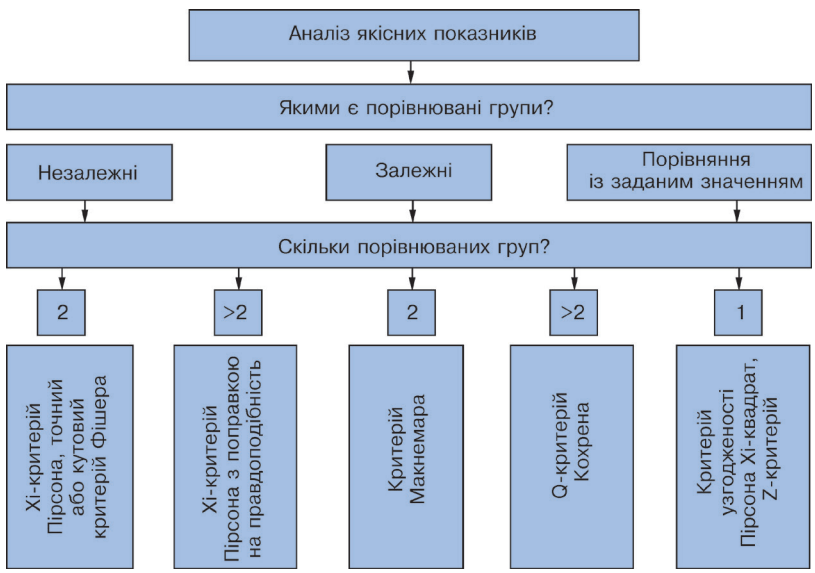



**Рисунок 2.6** – Етапи аналізу якісних даних у кіберспорті

На основі статистичних даних (вибірки) вирішують питання, чи нульову гіпотезу $H_0$ прийняти, чи відхилити на користь альтернативної гіпотези $H_1$.

**Статистичним критерієм гіпотези** (або просто критерієм гіпотези) називають випадкову величину $K$ (критерій), за допомогою якої проводиться перевірка гіпотези.

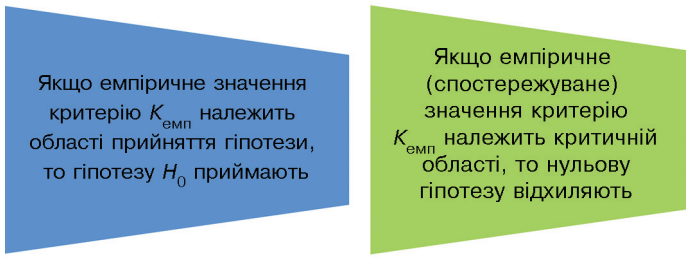
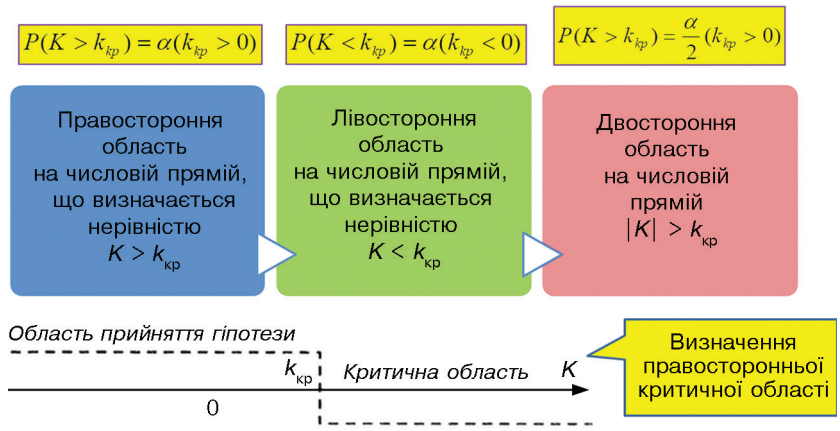



**Рисунок 2.7** – Правила перевірки статистичних гіпотез, де $K_{емп}$ – емпіричне (отримане в результаті розрахунків) значення критерію

**Рисунок 2.8** – Види критичних областей



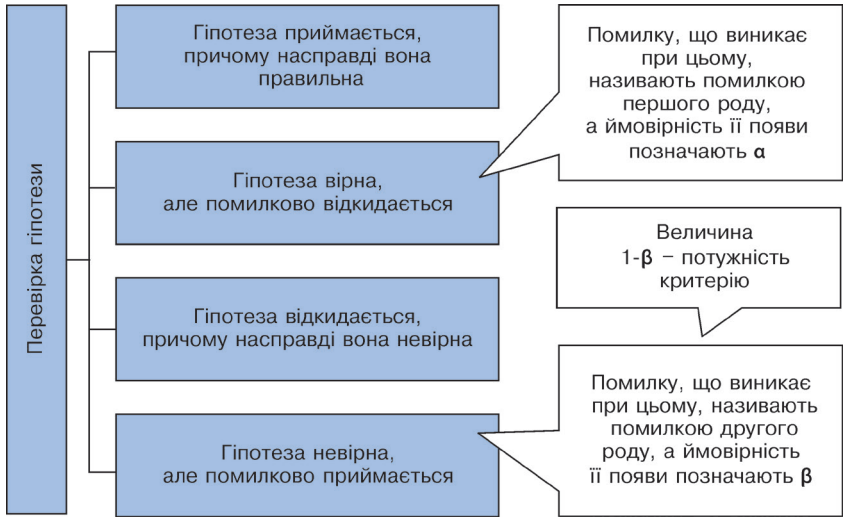




**Рисунок 2.9** – Можливі випадки при висуванні гіпотези

Сукупність значень критерію $K$, за яких нульова гіпотеза $H_0$ відхиляється, називають критичною областю, а сукупність значень критерію $K$, за яких нульову гіпотезу $H_0$ приймають, називають **областю прийняття гіпотези** (областю допустимих значень).

Систематизуємо правила перевірки статистичних гіпотез (рис. 2.7).

Табличні значення критеріїв знаходять за допомогою спеціальних статистичних таблиць або (для окремих критеріїв) за допомогою вбудованих функцій Excel.

Розглянемо види критичних областей (рис. 2.8) та можливі випадки при висуванні гіпотези (рис. 2.9).

## НОРМАЛЬНИЙ ЗАКОН РОЗПОДІЛУ

Однією з найбільш важливих задач статистичного аналізу є перевірка спостережуваних даних на підпорядкування **нормальному закону розподілу**. Як розглядалося вище (див. рис. 2.5), вирішення питання про вибір критерію для аналізу кількісних показників залежить від їхнього підпорядкування нормальному закону розподілу (рис. 2.10).

Закон розподілу ймовірностей безперервної випадкової величини X називають **нормальним**, якщо функція щільності її ймовірності описується формулою Лапласа–Гауса;


Формула складається з двох математичних констант: число пі наближено рівне 3,142; *e* – основа натурального логарифму, 2,718; *a* і σ – параметри розпо-

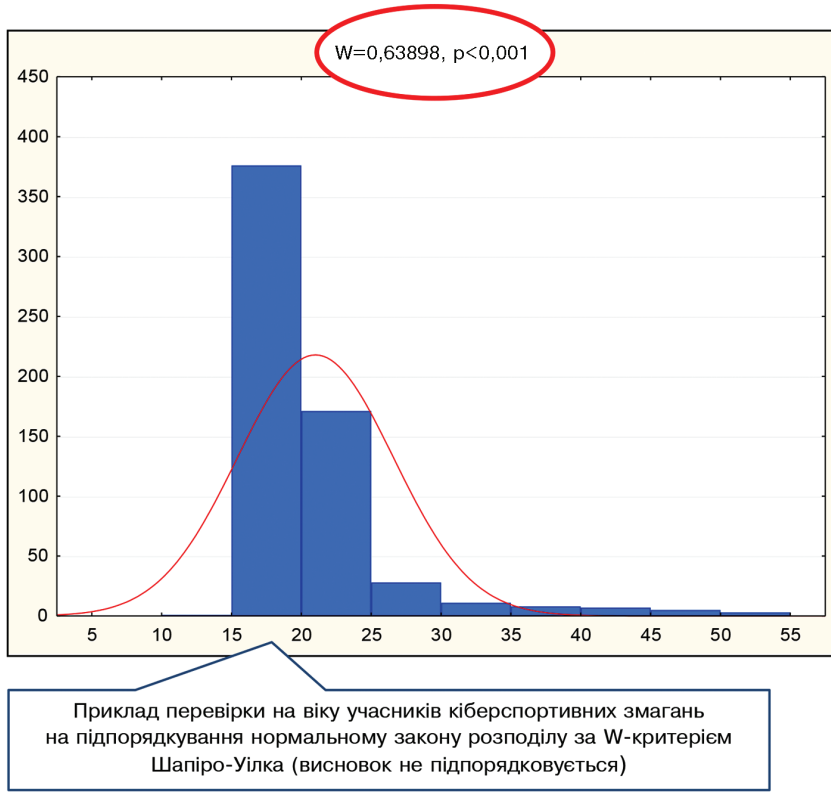



**Рисунок 2.10** – Перевірка показників на відповідність нормальному закону розподілу

ділу: *a* – математичне сподівання, що визначає центр розподілу, якому відповідає максимальна висота графіка (у різних джерелах можуть використовуватись інші позначення, наприклад, m або μ); $\sigma^2$ – дисперсія.

Іншими словами, закон розподілу показників називається нормальним, якщо більшість даних у вибірці концентрується біля середнього значення показника. У графічному представленні вибірка показників, які підпорядковуються нормальному закону розподілу, має форму дзвона.

Аналітичні ознаки нормального розподілу:

• У нормального розподілу математичне сподівання (вибіркова середня) збігається з модою та медіаною.

• Виконується правил «трьох сигм». Тобто, практично всі значення нормальної випадкової величини потрапляють у інтервал $(\alpha - 3 \cdot \sigma; \alpha + 3 \cdot \sigma)$.

• Коефіцієнти асиметрії A та ексцесу E нормального розподілу дорівнюють нулю.

Зауважимо, що асиметрія і ексцес – коефіцієнти, які характеризують його геометричну форму. **Асиметрія** характеризує міру скошеності графіка ліворуч/праворуч, а **ексцес** – міру його видовженості чи сплющеності.

Наближено перевірити відповідність форми розподілу нормальному закону розподілу можна на основі показників асиметрії A та ексцесу E.

Спостережувана величина розподілена за нормальним законом, якщо виконуються співвідношення:

$$|A| \le 3\sqrt{D(A)} \tag{2.1}$$

$$|E| \le 5\sqrt{D(E)} \tag{2.2},$$

де $D(A)$, $D(E)$ – дисперсії асиметрії і ексцесу відповідно, то розподіл вважається нормальним.

**Дисперсія** – це величина, яка характеризує ступінь розкиду кількісних показників відносно середнього значення.

При цьому зазначені дисперсії було обчислено за формулами:

$$D(A) = \frac{6(n-1)}{(n+1)(n+3)} \tag{2.3}$$

$$D(E) = \frac{24(n-2)(n-3)n}{(n-1)^2(n+3)(n+5)} \tag{2.4},$$

де $n$ – обсяг вибірки.

**Приклад 1.** *Постановка задачі. З*а даними офіційного сайту з кіберспорту наближено перевірити, чи підпорядковуються середня кількість глядачів (AMA або ACCV) та середня кількість каналів нормальному закону розподілу.

*Аналіз задачі.* Показниками популярності трансляції є **середня кількість глядачів** – кількість уболівальників, які одночасно спостерігають весь період

трансляції (AMA або ACCV), та **середня кількість каналів** – середня кількість одночасно активних каналів у категорії за визначений період часу (рис. 2.11).

Перевірити підпорядкування спостережуваних даних нормальному закону розподілу можна за допомогою критеріїв Пірсона, Колмогорова–Смирнова, Шапіро–Уілка. Одним із найпростіших методів для здійснення такої перевірки є метод наближеної перевірки відповідності даних нормальному закону розподілу за показниками ексцесу й асиметрії.

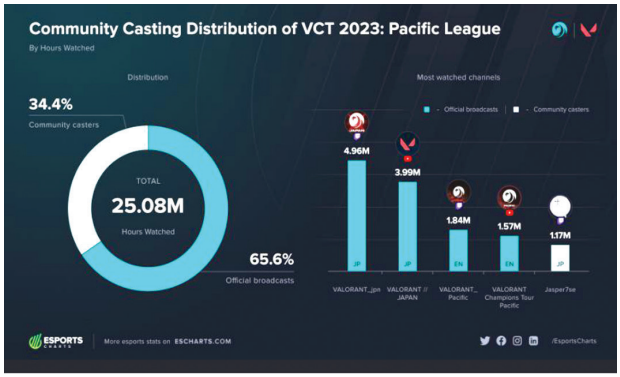




**Рисунок 2.11** – Середня кількість глядачів і каналів

*Хід роботи*

1. Внести початкові дані в MS Excel (рис. 2.12 (а)).

2. За допомогою надбудови «Аналіз даних» (Вставлення → Дані → Аналіз даних → Описова статистика) знайти асиметрію й ексцес показників (рис. 2.12 (б)).

3. Обчислити дисперсію асиметрії й ексцесу за формулами (2.3) і (2.4) (рис. 2.13).

4. Перевірити виконання умов (2.1) і (2.2) (рис. 2.14).

Акцентуємо увагу на тому, що питання підключення надбудови MS Excel «Аналіз даних» та приклади роботи з надбудовою докладно розглядалися у попередньому курсі «Інформаційні технології в кіберспорті (esports)».

***Висновок.*** Наближена перевірка відповідності розподілу середньої кількості глядачів підпорядковується нормальному закону розподілу, отже в середньому за досліджуваний період кількість глядачів становила 2699608.

Щодо середніх каналів, то наближена перевірка відповідності розподілу середніх каналів не підпорядковується нормальному закону розподілу, отже середня кількості одночасно активних каналів у категорії за визначений період часу становила 1927995624 (дивимося значення медіани).

**Зауваження!** Середні показники, які підпорядковуються нормальному закону розподілу, представляють у вигляді середньої та стандартного відхилення *SD*, величина якого рівна кореню квадратному із дисперсії: ($\bar{x} \pm SD$) ум.од. У випадку, коли нормальний закон розподілу показників довести не вдалося,

| № | Дата | AMA | Середні канали |
|---|---|---|---|
| 1 | Січ.21 | 2 918 417 | 1 872 221 076 |
| 2 | Лют.21 | 2 943 446 | 2 162 546 512 |
| 3 | Бер.21 | 2 928 369 | 1 971 127 076 |
| 4 | Кві.21 | 3 099 202 | 2 172 361 608 |
| 5 | Тра.21 | 3 100 114 | 2 222 643 776 |
| 6 | Чер.21 | 2 724 583 | 2 299 250 460 |
| 7 | Лип.21 | 2 626 124 | 1 948 530 600 |
| 8 | Сер.21 | 2 654 084 | 1 953 835 920 |
| 9 | Вер.21 | 2 573 103 | 1 967 560 692 |
| 10 | Жов.21 | 2 707 758 | 1 837 623 884 |
| 11 | Лис.21 | 2 515 253 | 1 986 591 792 |
| 12 | Гру.21 | 2 526 675 | 1 764 449 960 |
| 13 | Січ.22 | 2 914 608 | 1 749 301 326 |
| 14 | Лют.22 | 2 938 355 | 2 161 181 292 |
| 15 | Бер.22 | 2 596 628 | 1 883 485 580 |
| 16 | Кві.22 | 2 583 320 | 1 927 995 624 |
| 17 | Тра.22 | 2 532 774 | 1 854 823 428 |
| 18 | Чер.22 | 2 540 971 | 1 881 428 968 |
| 19 | Лип.22 | 2 535 680 | 1 823 993 688 |
| 20 | Сер.22 | 2 609 537 | 1 884 854 708 |
| 21 | Вер.22 | 2 506 029 | 1 939 755 464 |
| 22 | Жов.22 | 2 515 356 | 1 803 922 948 |
| 23 | Лис.22 | 2 500 609 | 1 858 009 320 |

Вхідні дані

*а*

**Рисунок 2.12** – Описова статистика для вхідного набору даних

| Середня кількість глядачів | | Середні канали | |
|---|---|---|---|
| Середнє | 2699608 | Середнє | 1916308960 |
| Стандартна помилка | 42037,17 | Стандартна помилка | 51322032,3 |
| Медіана | 2609537 | Медіана | 1927995624 |
| Стандартне відхилення | 201603,18 | Стандартне відхилення | 246131820,3 |
| Дисперсія вибірки | 40643844084,44 | Дисперсія вибірки | 6,05809E+16 |
| Ексцес | -0,78 | Ексцес | 7,765155726 |
| Асиметрія | 0,83 | Асиметрія | -2,002628883 |
| Інтервал | 599505 | Інтервал | 1279419012 |
| Мінімум | 2500609 | Мінімум | 1019831448 |
| Максимум | 3100114 | Максимум | 2299250460 |
| Сума | 62090995 | Сума | 44075106074 |
| Рахунок | 23 | Рахунок | 23 |

Результат розрахунків

*б*

| Середня кількість глядачів | | Середні канали | |
|---|---|---|---|
| Середнє | 2699608 | Середнє | 1916308960 |
| Медіана | 2609537 | Медіана | 1927995624 |
| Стандартне відхилення | 201603,18 | Стандартне відхилення | 246131820,3 |
| Ексцес | -0,78 | Ексцес | 7,76 |
| Асиметрія | 0,83 | Асиметрія | -2,00 |
| Рахунок | 23 | Рахунок | 23 |
| D(A) | =6*(F16 −1)/(F16 +1)/(F16+3) | | |
| D(E) | | D(E) | |

=24*(F16 −2)* (F16−3)*F16/(F16/−1)^2/*(F16 +3)/ (F16 +5)

Внесемо формули в комірки MS Excel, де F16 – посилання на комірку з величиною обсягу вибірки (Рахунок)

Так само, за допомогою формули, обчислимо D(E)

**Рисунок 2.13** – Розрахунок дисперсії асиметрії та ексцесу


**Рисунок 2.14** – Перевірка виконання умов підпорядкування даних нормальному закону розподілу

для оприлюднення середнього використовують медіану Me та 25 і 75 процентилі: Me (25 %; 75 %) ум. од.

25 і 75 процентилі можна розрахувати за допомогою вбудованої статистичної функції Excel ПРОЦЕНТИЛЬ.ВКЛ (PERCENTILE.INC).

## ВИЯВЛЕННЯ ТЕНДЕНЦІЙ У РОЗВИТКУ КІБЕРСПОРТУ

### Принципи застосування непараметричних критеріїв

Нагадаємо, що залежно від підпорядкування даних нормальному розподілу, обирається статистичний критерій – параметричний або непараметричний.

Слід зазначити, що ступінь розробленості питання, як здійснювати порівняльний аналіз вибіркових сукупностей за допомогою параметричних критеріїв набагато більша порівняно з непараметричними критеріями. І тут особливої популярності серед дослідників здобув t-критерій Стьюдента. Утім, як відомо, застосування параметричних критеріїв передбачає виконання ряду умов, серед яких – умова нормальності розподілу кожної з порівнюваних вибіркових сукупностей. Задачу здійснення аналізу даних із використанням параметричних критеріїв реалізовано в програмі MS Excel за допомогою надбудови MS «Excel Аналіз» даних. Такі задачі було розглянуто в попередньому курсі.

Як показує накопичений досвід, під час аналізу даних у сфері кіберспорту, особливо коли йдеться про невеликі за обсягом вибірки, емпіричні дані не підпорядковуються нормальному закону розподілу. У таких випадках на допомогу приходять непараметричні методи оцінки та аналізу статистичних гіпотез.

Застосування параметричних критеріїв (t-критерію Стьюдента) передбачає виконання умови нормальності розподілу кожної з порівнюваних вибіркових сукупностей та умову рівності вибіркових дисперсій

Непараметричні методи аналізу даних – обчислювальні процедури, під час використання яких застосовують не параметри розподілу, а результати ранжування

Критерій Манна-Уітні являє собою непараметричну альтернативу t-критерію Стьюдента для незалежних вибіркових сукупностей; Т-критерій Вілкоксона – непараметрична альтернатива t-критерію Стьюдента для залежних вибіркових сукупностей

**Рисунок 2.15** – Принципи застосування непараметричних критеріїв


**Рисунок 2.16** – Умови застосування U-критерію Манна–Уітні

**Непараметричні методи** перевірки статистичних гіпотез – це обчислювальні процедури, під час використання яких не застосовують параметри розподілу, а використовують результати ранжування або підрахунку значень досліджуваної ознаки.

Варто акцентувати увагу на тому, що надбудова MS «Excel Аналіз» даних *не дозволяє* автоматизувати процес розрахунку непараметричних критеріїв.

Розглянемо принципи застосування непараметричних критеріїв (рис. 2.15).

Для порівняння двох незалежних вибірок (наприклад, дві команди кіберспортсменів різного рівня технічної підготовленості; дві команди кіберспортсменів, які тренувалися за різними програмами; вибірки кіберспортсменів, які виступають у різних кіберспортивних дисциплінах; середня кількість глядачів за 2022 і 2023 рр. тощо) використовується непараметричний **U-критерій Манна–Уітні**.

Критерій Манна–Уітні являє собою непараметричну альтернативу t–критерію Стьюдента для незалежних вибіркових сукупностей. Перевага його полягає в тому, що застосування не вимагає припущень щодо нормальності розподілу й однакових дисперсій.

Метод полягає у визначенні того, чи є зона значень між двома варіаційними рядами, які перехрещуються достатньо, малою. Чим менше значення критерію, тим імовірніше, що відмінності між значеннями параметра у вибірках суттєві.

Перевага цього критерію полягає в тому, що його застосування не вимагає припущень щодо нормальності розподілу й однакових дисперсій.

Умови застосування U-критерію Манна–Уітні представлено на рисунку 2.16.

У разі застосування U-критерію Манна–Уітні висувають нульову та альтернативну до неї гіпотези:

- $H_0$ {Рівень ознаки у вибірці 2 не нижчий за рівень ознаки у вибірці 1}.
- $H_1$ {Рівень ознаки у вибірці 2 нижчий за рівень ознаки у вибірці 1}.

Після цього використовують порядок дій, представлений на рисунку рис. 2.17.

Зазначимо, що $\mathbf{n_1}$ і $\mathbf{n_2}$ – обсяги вибірок (кількість спостережуваних показників). Також у формулі розрахунку фігурують $\mathbf{n_x}$ – кількість елементів більшої за обсягом вибірки та $\mathbf{T_x}$ – більша з двох рангових сум.

Критичне значення U-критерію Манна–Уітні знаходять за допомогою статистичних таблиць. Воно залежить від обсягів вибірок, які порівнюються між собою. Число на перетині розміру найбільшої і найменшої вибірки є критичним значенням U-критерію Манна–Уітні на обраному рівні значущості $\alpha$.

Порядок розрахунку кількості елементів більшої за обсягом вибірки $n_x$ та більшої з двох рангових сум $T_x$ представлено на рисунку 2.18.

**Приклад 2.** *Постановка задачі.* Порівняти рівень розумової працездатності кіберспортсменів залежно від кіберспортивної дисципліни (дисципліна I, дисципліна II), якщо результати попередньої перевірки показали, що вибірки не підпорядковуються нормальному закону розподілу. Результати спостережень представлено в табличному вигляді.

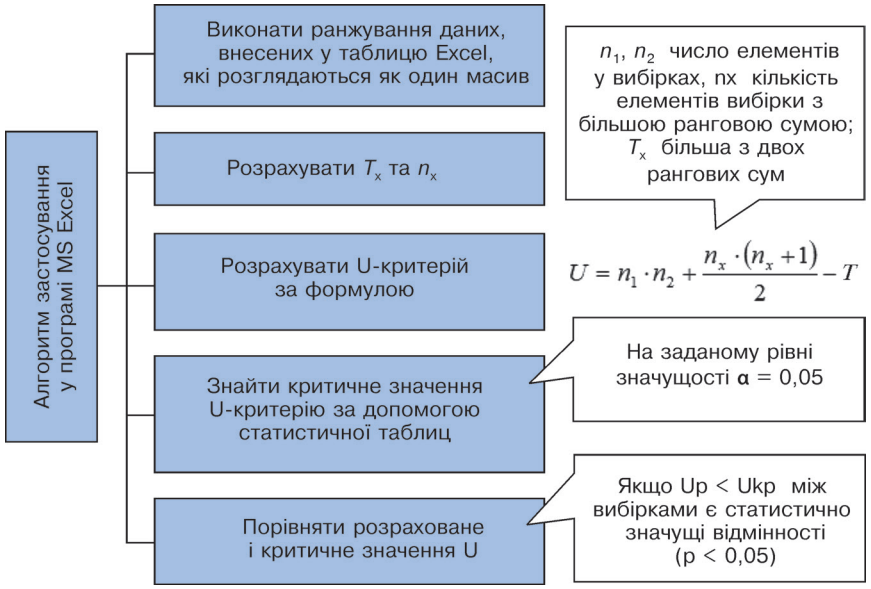



**Рисунок 2.17** – Порядок застосування U-критерію Манна–Уітні у програмі MS Excel

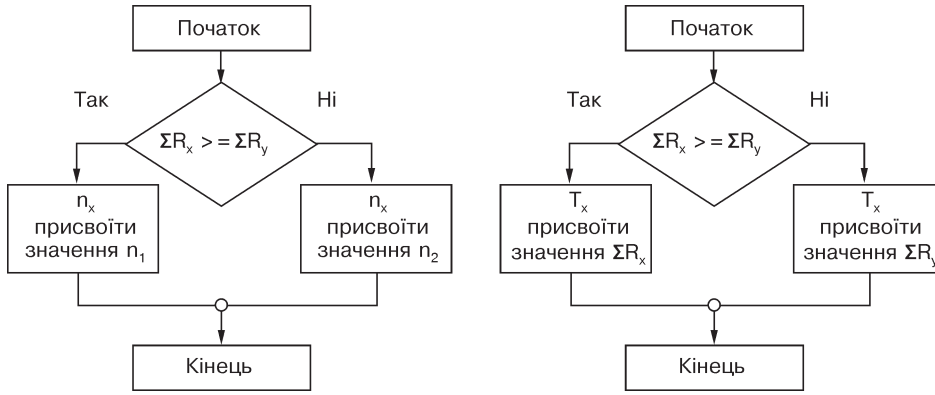



**Рисунок 2.18** – Порядок розрахунку кількості елементів більшої за обсягом вибірки та більшої з двох рангових сум

| C | D |
|---|---|
| Age | |
| I | II |
| 838 | 701 |
| 643 | 645 |
| 583 | 702 |
| 709 | 803 |
| 538 | 800 |
| 714 | 710 |
| 778 | 680 |
| 903 | 820 |
| 741 | 681 |
| 1020 | 700 |
| 1040 | 730 |
| 1020 | 620 |
| 799 | 800 |
| 685 | 901 |
| 626 | 703 |
| 830 | 742 |
| 848 | |
| 854 | |
| 882 | |
| 852 | |

*Аналіз задачі.* Маємо дві вибірки. Оскільки кіберспортсмени спеціалізуються у різних дисциплінах, то вибірки – незалежні.

Оскільки вибірки не підпорядковуються нормальному закону розподілу, для порівняльного аналізу слід застосувати непараметричний U-критерій Манна–Уітні.

*Хід роботи*

1. Внести початкові дані в MS Excel.
2. Внести відповідні формули та функції у комірки (рис. 2.19).

Зверніть увагу, що кожному показнику присвоєно ранг за допомогою статистичної функції РАНГ.СР за даними обох вибірок. Масив слід виділити в комірці «Посилання» та натиснути клавішу F4, завдяки чому масив буде зафіксованим.

Розрахунки обсягу вибірок та їхніх рангових сум здійснюють за допомогою відповідних вбудованих математичних функцій РАХУНОК та СУМА. Решта користувацьких формул вводиться з клавіатури.

3. Знайти табличне значення критерію (рис. 2.20).
4. Зробити висновки.

Варто акцентувати увагу на тому, що U-критерій Манна–Уітні переважно використовується для невеликих за обсягом вибірок. Статистична таблиця для знаходження його критичного значення передбачає, що обсяг вибірок не перевищує 61. У тому випадку, коли обсяг однієї (або обох) вибірок перевищує 60, розподіл U-статистики Манна–Уітні наближається до нормального розподілу. Тоді для перевірки гіпотези $H_0$ про відсутність відмінностей між вибірками використовується Z-статистика, де величина Z обчислюється за формулою:

$$Z = \frac{U - \frac{n_1 n_2}{2}}{\sqrt{\frac{n_1 n_2 (n_1 + n_2 + 1)}{12}}} \qquad (2.5)$$

| C | D | E | F | G | H |
|---|---|---|---|---|---|
| Age | | $R_x$ | $R_y$ | | |
| I | II | | | | |
| 838 | 701 | 27 | 11 | | |
| 643 | 645 | 5 | 6 | =RANK.AVG(D3;$C$3:$D$22;1) | |
| 583 | 702 | 2 | 12 | | |
| 709 | 803 | 14 | 24 | | |
| 538 | 800 | 1 | 22,5 | | |
| 714 | 710 | 16 | 15 | | |
| 778 | 680 | 20 | 7 | p | 0,05 |
| 903 | 820 | 33 | 25 | | |
| 741 | 681 | 18 | 8 | $n_1$ | 20 |
| 1020 | 700 | 34,5 | 10 | $n_2$ | 16 |
| 1040 | 730 | 36 | 17 | | |
| 1020 | 620 | 34,5 | 3 | $\sum R_x$ | 419 |
| 799 | 800 | 21 | 22,5 | $\sum R_y$ | 247 |
| 685 | 901 | 9 | 32 | | |
| 626 | 703 | 4 | 13 | $n_x$ | 20 |
| 830 | 742 | 26 | 19 | $T_x$ | 419 |
| 848 | | 28 | | | |
| 854 | | 30 | | U | 111 |
| 882 | | 31 | | $U_{cr}(0,05;20;16)$ | 98 |
| 852 | | 29 | | result | H0 |

Оскільки U>Uкр, то приймається гіпотеза H0, тобто статистично значущих відмінностей між розумовою працездатністю кіберспортсменів залежно від їх спеціалізації не доведено (p>0,05)

=COUNT(C3:C22)

=SUM(E3:E22)

=IF(H11>H12;H11;H12)

=IF(H14>H15;H14;H15)

=H11*H12+H17*(H17+1)/2-H18

=IF(H20>H21;"H0";"H1")

**Рисунок 2.19** – Приклад реалізації порівняльного аналізу за U-критерієм Манна–Уітні у програмі Excel

| $N_1$ | $N^2$ | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** |
| **3** | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 5 | 5 | 6 | 6 | 7 |
| **4** | 3 | 4 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 11 | 12 |
| **5** | 5 | 6 | 7 | 8 | 9 | 11 | 12 | 13 | 14 | 15 | 17 | 18 |
| **6** | 6 | 8 | 10 | 11 | 13 | 14 | 16 | 17 | 19 | 21 | 22 | 24 |
| **7** | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 |
| **8** | 10 | 13 | 15 | 17 | 19 | 22 | 24 | 26 | 29 | 31 | 34 | 36 |
| **9** | 12 | 15 | 17 | 20 | 23 | 26 | 28 | 30 | 34 | 37 | 39 | 42 |
| **10** | 14 | 17 | 20 | 23 | 26 | 29 | 33 | 36 | 39 | 42 | 45 | 48 |
| **11** | 16 | 19 | 23 | 26 | 30 | 33 | 37 | 40 | 44 | 48 | 51 | 55 |
| **12** | 18 | 22 | 26 | 29 | 33 | 37 | 41 | 45 | 49 | 53 | 57 | 61 |

**Критичне значення** U-критерію знайдемо за допомогою статистичної таблиці, де рівень значущості α приймемо за 0,05, а за $n_i$ – обсяги вибірок, де i=1,2

Число на перетині розміру найбільшої і найменшої вибірки є критичним значенням U-критерію Манна-Уітні

Uкр (10; 8; 0,05)=17

**Рисунок 2.20** – Порядок знаходження критичного значення U-критерію Манна–Уітні

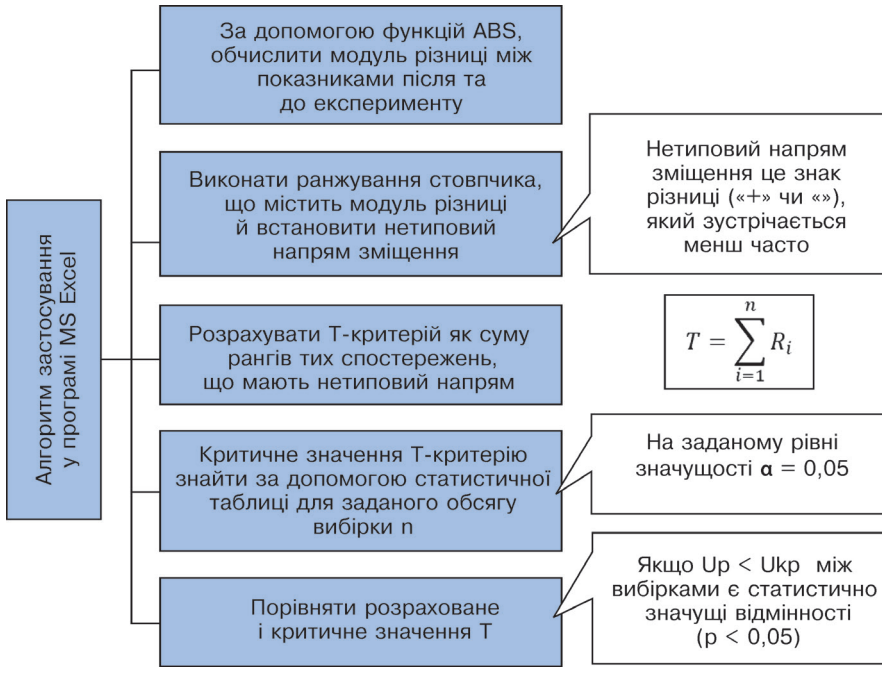



**Рисунок 2.21** – Порядок реалізації порівняльного аналізу за Т-критерієм Вілкоксона

Тоді, згідно зі статистичними таблицями, критичне значення Z (із яким слід порівнювати розраховане Z), приймається рівним ± 1,96.

Для зіставлення показників, виміряних у двох різних умовах на одній і тій самій вибірці випробуваних, показники яких не відповідають нормальному закону розподілу, використовується непараметричний **Т-критерій Вілкоксона**.

Цей критерій можна застосувати в тих випадках, коли ознаки виміряні принаймні у порядковій шкалі, і зрушення між другим і першим вимірами теж можуть бути впорядковані.

Застосування методу передбачає, що типовим зміщенням буде зрушення в напрямку, який зустрічається частіше, а нетиповим – зрушення в напрямку, який зустрічається рідко.

Головне обмеження – мінімальна кількість випробовуваних, які пройшли вимірювання в двох умовах – це 5 осіб.

Порядок здійснення порівняльного аналізу за Т-критерієм Вілкоксона представлено на рисунку 2.21.

**Зауваження!** Нагадаємо, що модуль числа – це відстань від початку відліку до точки, яка зображує це число.

**Приклад 3.** *Постановка задачі.* Дано показники розумової працездатності однієї групи кіберспортсменів до і після впровадження інноваційної програми тренування, спрямованої на підвищення їх розумової працездатності.

Встановити, чи мала запропонована технологія позитивний ефект (чи підвищилась розумова працездатність учасників дослідження).

Результати спостережень представлено в табличному вигляді.

| A | B | C |
|---|---|---|
| № | Розумова працездатність | |
| | До дослідження | Після дослідження |
| 1 | 838 | 830 |
| 2 | 643 | 750 |
| 3 | 583 | 690 |
| 4 | 709 | 810 |
| 5 | 538 | 600 |
| 6 | 714 | 700 |
| 7 | 778 | 750 |
| 8 | 903 | 1000 |
| 9 | 741 | 752 |
| 10 | 1020 | 1000 |
| 11 | 1040 | 1050 |
| 12 | 1020 | 1030 |
| 13 | 799 | 800 |
| 14 | 685 | 750 |
| 15 | 626 | 630 |
| 16 | 830 | 873 |
| 17 | 848 | 855 |
| 18 | 854 | 850 |
| 19 | 882 | 900 |
| 20 | 852 | 900 |

*Аналіз задачі.* Оскільки маємо дві вибірки одних і тих самих кіберспортсменів – до і після дослідження, то вибірки – залежні.

Оскільки вибірки не підпорядковуються нормальному закону розподілу, для порівняльного аналізу слід застосувати непараметричний Т-критерій Вілкоксона.

*Хід роботи*

1. Внести початкові дані в MS Excel.
2. Внести відповідні формули та функції у комірки (рис. 2.22).

Зверніть увагу, що нетиповими називають додатні або від'ємні різниці між показниками, які рідше зустрічаються. У нашому випадку, це від'ємні різниці.

3. Знайти табличне значення критерію (рис. 2.23).
4. Зробити висновки.

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | | Розумова працездатність | | | | |
| 2 | № | До дослідження | Після дослідження | Різниця (Після-До) | Модуль різниці | Ранг різниці |
| 3 | 1 | 838 | 830 | =C3-B3 | 8 | 5 |
| 4 | 2 | 643 | 750 | 107 | 107 | 19,5 |
| 5 | 3 | 583 | 690 | 107 | 107 | 19,5 |
| 6 | 4 | 709 | 810 | 101 | 101 | 18 |
| 7 | 5 | 538 | 600 | 62 | 62 | 15 |
| 8 | 6 | 714 | 700 | -14 | 14 | 9 |
| 9 | 7 | 778 | 750 | -28 | 28 | 12 |
| 10 | 8 | 903 | 1000 | 97 | 97 | 17 |
| 11 | 9 | 741 | 752 | 11 | 11 | 8 |
| 12 | 10 | 1020 | 1000 | -20 | 20 | 11 |
| 13 | 11 | 1040 | 1050 | 10 | 10 | 6,5 |
| 14 | 12 | 1020 | 1030 | 10 | 10 | 6,5 |
| 15 | 13 | 799 | 800 | 1 | 1 | 1 |
| 16 | 14 | 685 | 750 | 65 | 65 | 16 |
| 17 | 15 | 626 | 630 | 4 | 4 | 2,5 |
| 18 | 16 | 830 | 873 | 43 | 43 | 13 |
| 19 | 17 | 848 | 855 | 7 | 7 | 4 |
| 20 | 18 | 854 | 850 | -4 | 4 | 2,5 |
| 21 | 19 | 882 | 900 | 18 | 18 | 10 |
| 22 | 20 | 852 | 900 | 48 | 48 | 14 |

Для обчислення модуля різниці використовуємо функцію ABS

Для обчислення рангу модуля різниці використовуємо функцію РАНГ.СР

Шукаємо нетипові зміщення, розглядаючи стовпчик «Різниця». Нетипові зміщення виділено червоним

Обчислюємо Т-критерій як суму рангів нетипових зміщень: Т=9+12+11+2,5=34,5

**Рисунок 2.22** – Приклад реалізації порівняльного аналізу за Т-критерієм Вілкоксона у програмі Excel

| Для обчислення модуля різниці | | | |
|---|---|---|---|
| | 0.025 | 0.01 | 0.005 |
| Шукаємо нетипові зсуви, | | | |
| N | 0.05 | 0.02 | 0.01 |
| 6 | 0 | - | - |
| 7 | 2 | 0 | - |
| 8 | 4 | 2 | 0 |
| 9 | 6 | 3 | 2 |
| 10 | 8 | 5 | 3 |
| 11 | 11 | 7 | 5 |
| 12 | 14 | 10 | 7 |
| 13 | 17 | 13 | 10 |
| 14 | 21 | 16 | 13 |
| 15 | 25 | 20 | 16 |
| 16 | 30 | 24 | 20 |
| 17 | 35 | 28 | 23 |
| 18 | 40 | 33 | 28 |
| 19 | 46 | 38 | 32 |
| 20 | 52 | 43 | 38 |
| 21 | 59 | 49 | 43 |
| 22 | 66 | 56 | 49 |
| 23 | 73 | 62 | 55 |
| 24 | 81 | 69 | 61 |

Приклад перевірки на вік учасників кіберспортивних змагань на підпорядкування нормальному закону розподілу за W-критерієм Шапіро-Уілка (висновок, не підпорядковується)

Число на перетині розміру найбільшої вибірки і найменшої вибірки є критичним значенням Т-критерію Манна-Уітні

**Критичне значення** Т-критерію знайдемо за допомогою статистичної таблиці, де рівень значущості α приймемо за 0,05, а за $n_i$ – обсяги вибірок, де i = 1,2

**Рисунок 2.23** – Порядок знаходження критичного значення Т–критерію Вілкоксона та формування висновків

| Група | |
|---|---|
| I | II |
| 838 | 701 |
| 643 | 751 |
| 583 | 852 |
| 709 | 803 |
| 538 | 800 |
| 714 | 710 |
| 778 | 980 |
| 903 | 820 |
| 741 | 881 |
| 720 | 700 |
| 740 | 730 |
| 720 | 620 |
| 799 | 800 |
| 685 | 901 |
| 626 | 703 |
| 830 | 742 |
| 848 | 1045 |
| 854 | 1045 |
| 882 | 1050 |
| 852 | 1051 |
| | 1052 |
| | 1053 |

## ЗАВДАННЯ ДО ПРАКТИЧНИХ РОБІТ

**Практична робота № 2.**

Порівняти рівень розумової працездатності кіберспортсменів залежно від кіберспортивної дисципліни (дисципліна I, дисципліна II), якщо результати попередньої перевірки показали, що вибірки не підпорядковуються нормальному закону розподілу.

Результати спостережень представлено в табличному вигляді.

**Зауваження.** Дивитися приклад 2.

Критичне значення U-критерію Манна–Уітні для вибірок $n_1 = 20$. $n_2 = 22$ становить 141:

$$\mathbf{U_{кр}(0{,}05;22;20) = 141}$$

**Практична робота № 3.**

Провели дослідження, під час якого команду з 10 кіберспортсменів перевірили на стресостійкість за методикою визначення нервово-психічної стійкості, ризику дезадаптації у стресі «Прогноз». З'ясувалося, що в цілому кіберспортсмени характеризуються задовільним рівнем стресостійкості. Після застосування технології посилення стресостійкості кіберспортсменів обстежили повторно. Встановити, чи мала технологія позитивний ефект (чи покращилися оцінки стресостійкості кіберспортсменів).

| Етап дослідження | |
|---|---|
| до | після |
| 14 | 12 |
| 20 | 17 |
| 18 | 15 |
| 22 | 21 |
| 11 | 12 |
| 10 | 10 |
| 21 | 17 |
| 12 | 12 |
| 15 | 14 |
| 20 | 15 |

## ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ОПРАЦЮВАННЯ

**Завдання 1.**

Вивчено дані 10 професійних українських кіберспортсменів та ТОП-5 гравців світу, які мають найбільшу кількість підписаних на їхні оновлення користувачів на Twitch (відеостримінговий сервіс, який спеціалізується на тематиці комп'ютерних ігор). Визначити, чи існують статистично значущі відмінності на рівні значущості $\alpha = 0{,}05$ ($p < 0{,}05$) між українськими і світовими кіберспортсменами за кількістю підписаних користувачів. Початкові дані наведено в таблиці 2.1.

**Таблиця 2.1** – Найбільш популярні за кількістю підписаних користувачів кіберспортсмени

| Українські | | Зарубіжні | |
|---|---|---|---|
| Nickname | Кількість підписаних користувачів, тис. | Nickname | Кількість підписаних користувачів, тис. |
| Edward | 23,8 | summit1g | 6200 |
| General | 58,5 | TimTheTatman | 7000 |
| Lil | 117 | Shroud | 10 500 |
| Always-wannafly | 118 | xQc | 11 800 |
| Yozhyk | 135,7 | Ninja | 18 500 |
| Resolut1on | 160,9 | | |
| No[o]ne | 318,8 | | |
| Iceberg | 369,4 | | |
| Dendi | 1 000 | | |
| S1mple | 3400 | | |

**Завдання 2.**

За даними спеціалізованих сайтів із кіберспорту визначити величину 10 максимальних призових за минулий і поточний роки з кібердисицпліни, якою ви займаєтесь. Встановити, чи існують між ними відмінності.

## КОНТРОЛЬНІ ЗАПИТАННЯ ТА ЗАВДАННЯ

***1. Як можна представити зв'язок між вимірюваннями і знаннями?***

***2. Які типи вимірювальних шкал вам відомі?***

***3. Назвіть характерні особливості номінальної шкали. Наведіть приклади показників у сфері кіберспорту, виміряних у номінальній шкалі.***

***4. Чим порядкова шкала вимірювання відрізняється від номінальної?***

***5. Яка шкала вимірювання є найбільш потужною?***

***6. Який аналіз даних можна вважати успішним?***

***7. Що являє собою вибірка? Наведіть приклади вибірок у сфері кіберспорту.***

***8. Яку вибірку вважають репрезентативною?***

***9. Чим відрізняються залежні і незалежні вибірки? Наведіть приклади залежних і незалежних вибірок у сфері кіберспорту.***

***10. Які види гіпотез вам відомі? Назвіть правила перевірки статистичних гіпотез.***

***11. Які види статистичного аналізу застосовують у сфері кіберспорту?***

***12. Який розподіл показників вважається нормальним?***

***13. Назвіть аналітичні ознаки нормального розподілу.***

***14. Для чого у сфері кіберспорту вихідні дані перевіряють на підпорядкування нормальному закону розподілу?***

***15. Що означає запис «$p < 0,05$»?***

***16. Назвіть принципи застосування непараметричних критеріїв.***

***17. Що являє собою статистичний критерій гіпотези?***

***18. Які непараметричні критерії вам відомі? У чому полягає принципова відмінність між ними?***

***19. Назвіть умови застосування Т-критерію Вілкоксона?***

***20. Як представляють вибіркові середні залежно від підпорядкування даних нормальному закону розподілу?***

ТЕМА 3

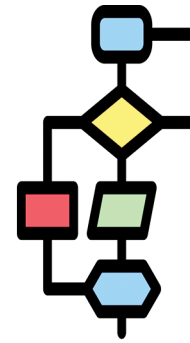
# АЛГОРИТМІЗАЦІЯ РОЗВ'ЯЗАННЯ ПРАКТИЧНИХ ЗАДАЧ У КІБЕРСПОРТІ

## ЛЕКЦІЯ 4. АЛГОРИТМІЗАЦІЯ ЯК МОДЕЛЬ РОЗВ'ЯЗАННЯ ПРАКТИЧНИХ ЗАДАЧ У СФЕРІ КІБЕРСПОРТУ

1. ***Основні поняття алгоритмізації.***
2. ***Способи запису алгоритмів.***
3. ***Складання блок-схем.***
4. ***Розробка лінійних алгоритмів.***
5. ***Застосування розгалужених і циклічних алгоритмів у ході розв'язання задач у сфері кіберспорту.***

### ОСНОВНІ ПОНЯТТЯ АЛГОРИТМІЗАЦІЇ

Започаткування та розвиток кіберспортивної науки є визначальним фактором для прогресивних суспільних перетворень. Наукова діяльність у сфері кіберспорту передбачає поетапний збір і аналіз даних, вивчення параметрів досліджуваних об'єктів, виявлення кореляційних взаємозв'язків між ними, встановлення закономірностей і тенденцій між об'єктами тощо. При цьому вимоги до точності вимірів і рівня обґрунтованості результатів досліджень продовжують зростати. Тому, як зазначалося в попередній темі, статистична обробка емпіричних даних є невід'ємною частиною діяльності науковця або управлінця у сфері кіберспорту. При цьому значне коло професійно орієнтованих завдань можна вирішитити за допомогою прикладного програмного забезпечення що унеможливлює появу технічних помилок при розрахунках. Утім на дослідника / управлінця накладається вимога чіткого усвідомлення порядку виконуваних ним дій для досягнення того чи іншого результату. У такому випадку мова йде про розробку алгоритму – алгоритмізацію виконуваних дій, тобто послідовне їх виконання.



З іншого боку, розробка прикладного програмного забезпечення окрім постановки задачі, її математичного опису та вибору методу розв'язання задачі передбачає алгоритмізацію розв'язання задачі, після чого власне й відбувається етап складання програми.

**Алгоритм** (процедура) – розв'язання задач у вигляді точних послідовно виконуваних вказівок (команд).

Відповідно, **комп'ютерна програма** – це алгоритм, записаний спеціальною алгоритмічною мовою програмування, який розуміє комп'ютер.

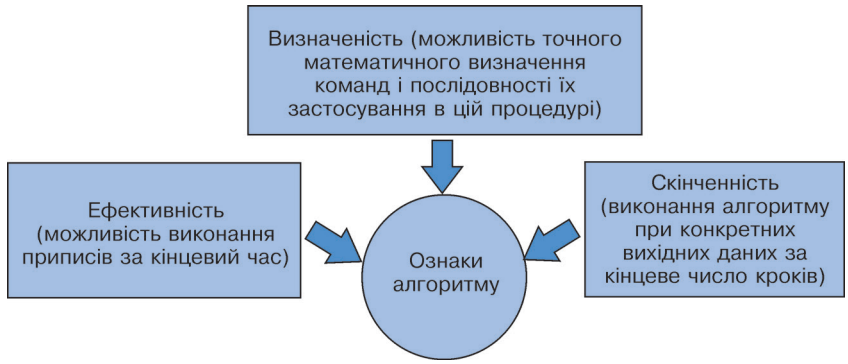


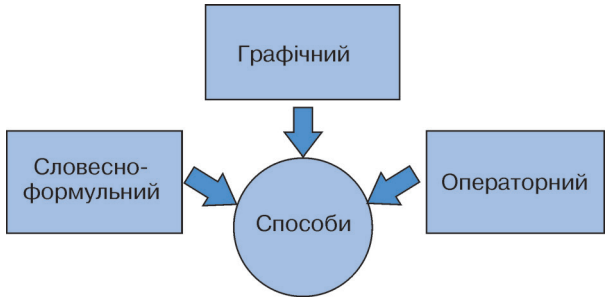

**Рисунок 3.1** – Властивості алгоритму

Отже, розробка алгоритмів є одним із основних етапів розв'язання задачі за допомогою комп'ютера або розробки комп'ютерної програми. Відповідно, виконавцем алгоритму є автоматичний пристрій (електронна обчислювальна машина, робот, верстат з числовим програмним управлінням) або користувач персонального комп'ютера (ПК), який виконує певний набір команд.

Прикладом алгоритму може слугувати інструкція для розв'язання задачі, поетапне представлення дій під час підготовки до турніру, графік тренувальних навантажень тощо.

Головними ознаками алгоритму є визначеність, ефективність та скінченність (рис. 3.1).

## СПОСОБИ ЗАПИСУ АЛГОРИТМІВ




Існують словесно-формульний, операторний та графічний способи представлення алгоритму (рис. 3.2).

**Рисунок 3.2** – Способи запису алгоритмів

## СКЛАДАННЯ БЛОК-СХЕМ

Найбільш інтуїтивно зрозумілим способом представлення алгоритму є **графічний,** який ще називають **блок-схемою** (рис. 3.3).

У блок-схемах використовують **геометричні фігури,** кожна з яких зображує якусь операцію або дію, а також етап процесу розв'язання задачі. Кожна фігура називається **блоком.** Порядок виконання етапів указується стрілками, що

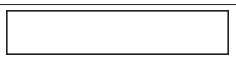
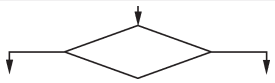
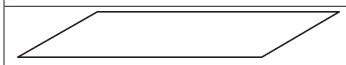
| Позначення | Виконувана функція |
|---|---|
|  | Початок чи кінець алгоритму |
|  | Обчислювальні дії |
|  | Перевірка умови: вибір одного з двох напрямів |
|  | Введення або виведення даних |
|  | Організація циклічних процесів |
|  | Напрямок ліній потоку – стрілки |

**Рисунок 3.3 –** Найчастіше вживані блоки

з'єднують блоки, які необхідно розміщувати зверху вниз або зліва направо в порядку їх виконання.

## Правила побудови алгоритмів на мові блок-схем

1. Блок-схема будується зверху вниз.
2. У будь-якій блок-схемі є один елемент, відповідний початку, і один елемент, відповідний закінченню.
3. Повинен бути хоча б один шлях з початку блок-схеми до будь-якого елемента.
4. Повинен бути хоча б один шлях від кожного елемента блок-схеми в кінець блок-схеми.

Для того, щоб комп'ютер міг виконати алгоритм, його треба написати зрозумілою для комп'ютера мовою. Комп'ютер розуміє машинну мову (у вигляді 0 і 1). Для того, щоб людина і комп'ютер розуміли один одного, розроблені спеціальні мови для записів алгоритмів – **алгоритмічні мови програмування.**

Алгоритмічна мова відрізняється від машинної мови тим, що складається зі слів і символів, як природна мова. У ній зазвичай 30–40 основних слів і дуже строгі правила складання речень. Основні слова алгоритмічної мови називають **службовими словами.** В алгоритмічних мовах використовують слова англійського алфавіту.

Алгоритм, який записаний на алгоритмічній мові, – це програма для комп'ютера. Кожне речення в програмі – **це оператор.**

Тому форму представлення алгоритму за допомогою операторів, які відображають порядок виконання дій за допомогою операторів, називають **операторним методом** опису алгоритмів. Його розроблено радянським ученим Олексієм Андрійовичем Ляпуновим у 1954 р.

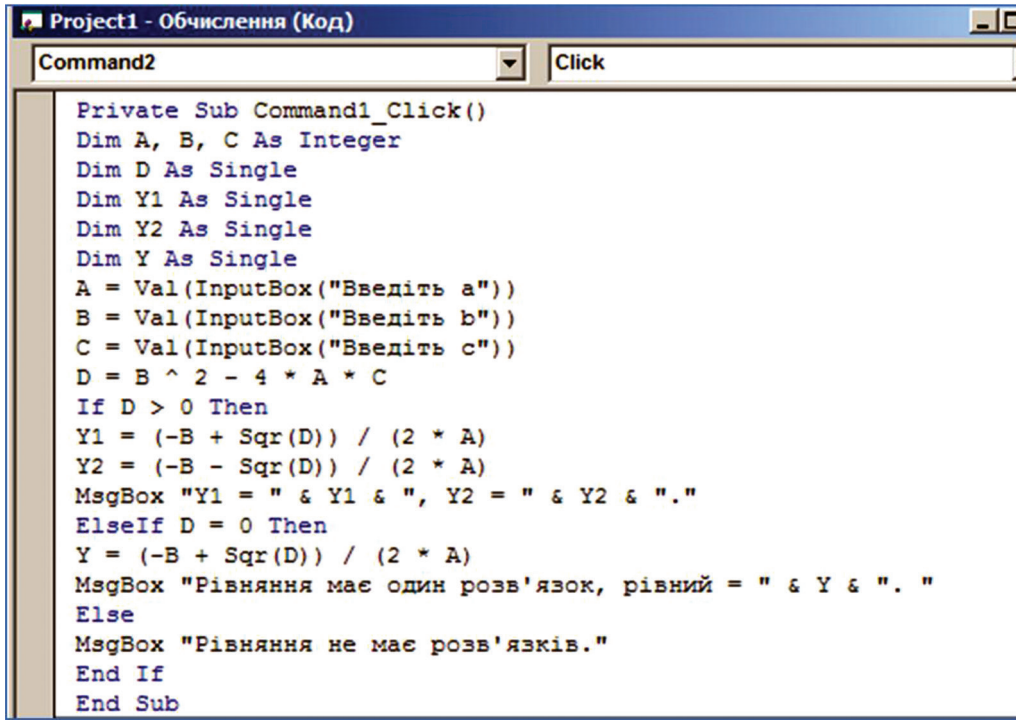



**Рисунок 3.4** – Код програми для розв'язання квадратного рівняння

Для представлення алгоритмів раніше вживали алгоритмічну мову АЛГОЛ-60 (ALGOL від ALGOrithmic Language – алгоритмічна мова, мова опису алгоритмів), яка також слугувала мовою програмування. Нині різниця між алгоритмічними мовами та мовами програмування стирається. Нагадаємо, що найвідомішими алгоритмічними мовами є Бейсік (Basic), Паскаль (Pascal), Фортран (Fortran) і сучасні мови, які розроблені на їхній основі (Pyton, Java, C++).

Приклад коду програми для розв'язання квадратного рівняння, представлений за допомогою алгоритмічної мови програмування Visual Basic, подано на рисунку 3.4.

## РОЗРОБКА ЛІНІЙНИХ АЛГОРИТМІВ

Алгоритми бувають лінійними, розгалуженими та циклічними (рис. 3.5).

**Лінійний алгоритм** – це алгоритм, в якому дії виконуються тільки один раз і строго в тому порядку, в якому вони записані. Це найпростіший тип алгоритмів, який дозволяє представити порядок послідовно виконуваних дій (рис. 3.6).

Прочитаємо наступний алгоритм (рис. 3.7).

Складемо алгоритм знаходження витрат **V** комп'ютерного клубу на оновлення 5 комп'ютерів **K** та 2 антивірусних програм **A**, де K – ціна за комп›ютер, A – ціна антивірусної програми (рис. 3.8).

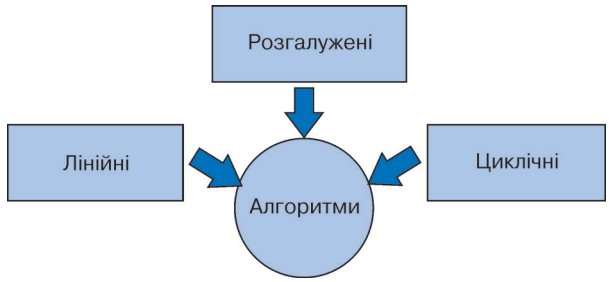


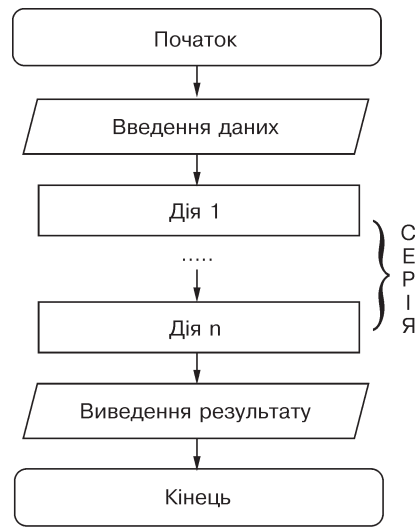

**Рисунок 3.5** – Типи алгоритмів

**алг** Назва алгоритму (опис аргументів та результатів)
- **арг** Перелік аргументів
- **рез** Перелік результатів

**поч** Опис допоміжних змінних
- Введення даних
- Дія 1
- Дія 2
- ·
- ·
- ·
- Дія n
- Вивід результату

**кін**

**Алгоритмічна мова**




**Блок-схема**

**Рисунок 3.6** – Загальний вигляд лінійного алгоритму на алгоритмічній мові та за допомогою блок-схеми

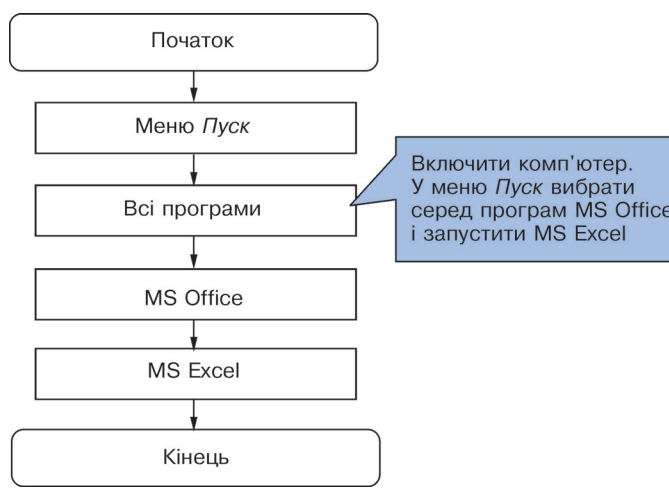



**Рисунок 3.7** – Алгоритм запуску програми MS Excel

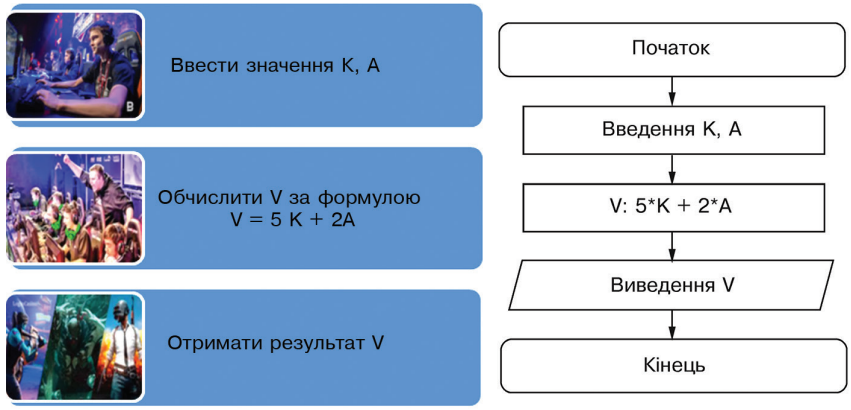



**Рисунок 3.8** – Алгоритм визначення витрат

**Приклад 1.** За вхідними даними було знайдено рівняння взаємозв'язку між кількістю турнірів із кіберспорту та кіберспортсменів, які брали у них участь. Рівняння регресії має такий вигляд:

$$y = 242{,}12 + 5{,}08 \cdot x \qquad (3.1),$$

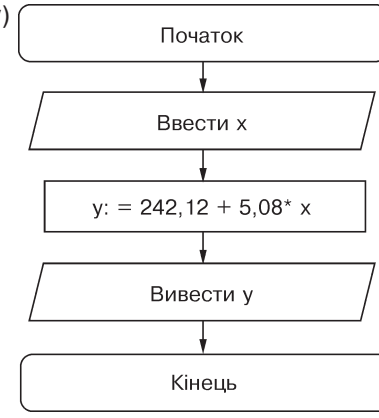
де $y$ – кількість турнірів; $x$ – кількість залучених гравців.

Скласти блок схему та алгоритм для прогнозування кількості турнірів за передбачуваною кількістю залучених гравців.

*Аналіз задачі.* Як видно з аналітичного представлення впливу кількості гравців на кількість організованих турнірів із кіберспорту, для визначення кількості турнірів потрібно вказати кількість залучених гравців. Тоді, відповідно до правил виконання арифметичних дій, спочатку розраховується добуток кількості гравців та коефіцієнта 5,08, а потім до отриманого результату додається вільний член регресії 242,12 (рис. 3.9).

*Результат розв'язання.*

```
алг Кількість турнірів (натуральне x, y)
    арг x
    рез y
поч
    Вивести «Введіть x»
    Введення x
    y: = 242,12 + 5,08*x
    Вивести y
кін
```




Алгоритмічна мова — Блок-схема

**Рисунок 3.9** – Алгоритм визначення кількості турнірів

**Приклад 2.** Дано результати регресійного аналізу засобами надбудови Аналіз даних програми MS Excel, що показують залежність між призовим фондом, кількістю гравців і кількістю турнірів із кіберспорту.

Скласти блок-схему та алгоритм розрахунку призового фонду за передбачуваною кількістю залучених гравців та кількістю турнірів.

**Результати регресійного аналізу**

| *Коефіцієнти* | |
|---|---|
| Y-пересічний | 2181177422, |
| Кількість гравців | −21563,25233 |
| Кількість турнірів | 90591,37094 |

*Аналіз задачі.* Згідно з результатами регресійного аналізу, аналітичне представлення впливу кількості гравців і кількості організованих турнірів із кіберспорту на величину призового фонду має такий вигляд:

$$y = 21811774{,}22 - 21563{,}25 \cdot x_1 + 90591{,}37 \cdot x_2 \qquad (3.2),$$

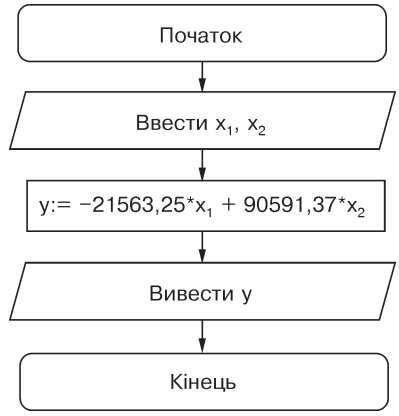
де $y$ – призовий фонд; $x_1$ – кількість турнірів; $x_2$ – кількість залучених гравців.

Після введення $x_1$ і $x_2$ розмір призового фонду розраховується відповідно до правил виконання арифметичних дій (рис. 3.10).

*Результат розв'язання.*

```
алг Розмір призового фонду (натуральне x1,
x2, y)
    арг x1, x2
    рез y
поч
    Вивести «Введіть x1»
    Введення x1
    Вивести «Введіть x2»
    y := −21563,25*x1 + 90591,37*x2
    Вивести y
кін
```

Алгоритмічна мова




Блок-схема

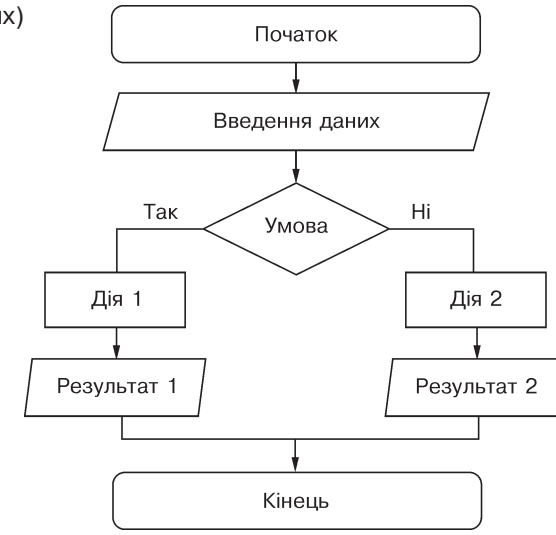
**Рисунок 3.10** – Алгоритм визначення розміру призового фонду

## ЗАСТОСУВАННЯ РОЗГАЛУЖЕНИХ І ЦИКЛІЧНИХ АЛГОРИТМІВ У ХОДІ РОЗВ'ЯЗАННЯ ЗАДАЧ У СФЕРІ КІБЕРСПОРТУ

**Команди розгалуження** – це складові командами, у яких присутні умови, залежно від істинності яких виконуються або не виконуються оператори, що входять до складу команди (рис. 3.11).

**алг** Назва алгоритму (опис змінних)
арг Перелік аргументів
рез Перелік результатів
**поч**
Введення даних
**якщо** Умова
**то** Дія 1
Виведення результату 1
**Інакше** Дія 2
Виведення результату 2
**кін**

**Алгоритмічна мова**




**Блок-схема**

**Рисунок 3.11** – Загальний вигляд розгалудженого алгоритму на алгоритмічній мові та за допомогою блок–схеми

**Чотири основні варіанти розгалуження**

- якщо – то
- якщо – то – інакше
- вибір
- вибір – інакше

Базова структура розгалуження залежно від результату перевірки умови (так або ні) забезпечує вибір одного з альтернативних шляхів роботи алгоритму.

Кожен шлях веде до загального виходу, тому робота алгоритму буде продовжуватися незалежно від того, який шлях буде обрано.

Приклад алгоритму, що перевіряє, чи проходить графік функції $y = 3x + 4$ через точку з координатами $(x_1, y_1)$ (рис. 3.12).

Наведемо приклади реалізації алгоритмів, що розгалужуються, у сфері кіберспорту. Якщо на турнірі кіберспортмен потрапить у трійку лідерів, то він отримає грошову винагороду або не отримає. Або, за умови кількох розгалужень, прикладом його реалізації може бути «якщо гравець буде тривалий час тренуватися і якщо при цьому в нього високий рівень стресостійкості, то він має шанс побудувати кар'єру кіберспортсмена або не має».

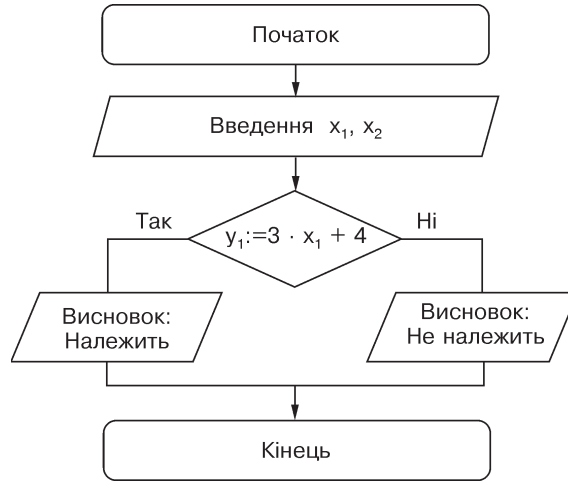



**Рисунок 3.12** – Блок-схема розгалуженого алгоритму

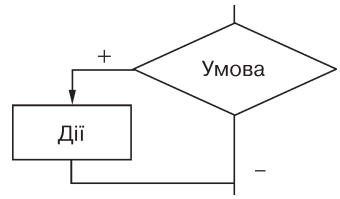
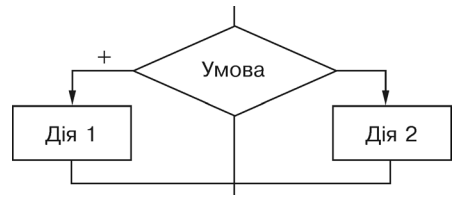
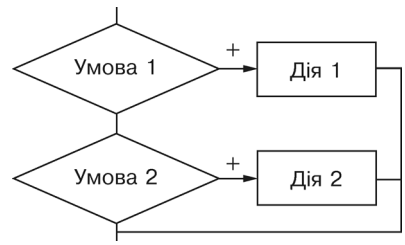
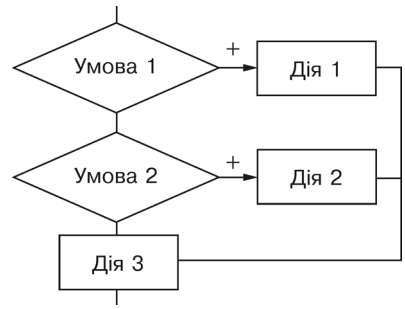
| Алгоритмічна мова | Блок-схема |
|---|---|
| **Конструкція «якщо – то» – неповне розгалуження** | |
| **якщо** умова<br>**то** дії<br>**все** |  |
| Конструкція «якщо – то – інакше» – повне розгалуження | |
| **якщо** умова<br>**то** дія 1<br>**інакше** дія 2<br>**все** |  |
| **Конструкція «вибір» – неповний вибір** | |
| **вибір**<br>**при** умова 1: дія 1<br>**при** умова 2: дія 2<br>**все** |  |
| Конструкція «якщо – то – інакше» – повне розгалуження | |
| **вибір**<br>**при** умова 1: дія 1<br>**при** умова 2: дія 2<br>інакше дія 3<br>**все** |  |

**Рисунок 3.13 –** Схема представлення розгалужених алгоритмів

Варіанти представлення розгалужених алгоритмів подано в рисунку 3.13.

Наведемо приклад алгоритму розв'язання квадратного рівняння $ax^2 + bx + c = 0$ в області дійсних чисел (рис. 3.14).


**Рисунок 3.14** – Алгоритм розв'язання квадратного рівняння

**Приклад 3.** Скласти алгоритм вибору гравця із трьох осіб, у якого найвищий рейтинг (рис. 3.15).

*Розв'язання.* Рейтинг Гравця 1 позначимо a, Гравця 2 – в, Гравця 3 – с.

**Словесний спосіб**

1. Ввести рейтинг гравців а, б, с
2. Проаналізувати, чи є рейтинг Гравця 1 більшим за рейтинг Гравця 2 та Гравця 3
3. Якщо так, то вивести повідомлення про вибір Гравця 1
4. Якщо ні , перевірити, чи є рейтинг Гравця 2 більшим за рейтинг Гравця 1 та Гравця 3
5. Якщо так, то вивести повідомлення про вибір Гравця 2
6. Якщо ні то вивести повідомлення про вибір Гравця 3

**Формульно-словесний**

1. Ввести a, b c
2. Якщо a > b, a > c, то вивести a
   Інакше
   Якщо b > a, b > c, то вивести b
   Інакше вивести c

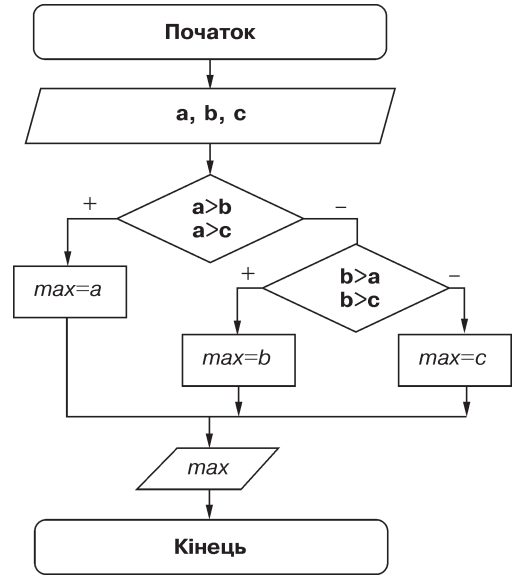



**Рисунок 3.15** – Алгоритм вибору гравця за рейтингом

Іншим прикладом розгалуженого алгоритму може слугувати визначення рівня гри кіберспортсмена. Зазначимо, що шкала для перевірки рівня гри в кіберспорті – це спосіб оцінки скілла та досвіду гравця у певній грі чи жанрі. Шкала може відрізнятися залежно від гри, але зазвичай вона грунтується на таких параметрах:

• *рейтинг гравця* – позиція у рейтинговій таблиці або лізі порівняно з іншими гравцями;

• *статистика гравця* – його результати у зіграних матчах (кількість перемог, поразок, вбивств, смертей, асистів, збитків, цілей тощо);

• *навички гравця* – вміння використовувати різні аспекти гри, такі як механіка, стратегія, тактика, комунікація, командна робота тощо.

**Приклад 4.** Побудувати алгоритм визначення рівня гри кіберспортсмена у дисципліні CS:GO, якщо шкала рівня гри має вигляд:

• Від 1 до 3 – низький рівень гри. Гравець має погане знання карт, зброї та гранат. Він часто робить помилки та не слухає команду. Його рейтинг знаходиться у срібній чи золотій лізі.

• Від 4 до 6 – середній рівень гри. Гравець має достатнє знання карт, зброї та гранат. Він іноді робить гарні ходи та слухає команду. Його рейтинг знаходиться у майстерній чи легендарній лізі.

• Від 7 до 9 – високий рівень гри. Гравець має відмінне знання карт, зброї та гранат. Він часто робить чудові ходи та координує команду. Його рейтинг перебуває в елітній чи глобальній лізі.

• 10 – професійний рівень гри. Гравець має експертне знання карт, зброї та гранат. Він постійно робить ефективні ходи та лідирує у команді. Його рейтинг знаходиться у топ–50 світу.

*Аналіз задачі.* Вочевидь, маємо задачу побудови розгалуженого алгоритму, який передбачає виконання трьох умов, відповідно до шкали визначення рівня гри кіберспортсмена.

*Розв'язання.* Оцінку гри позначимо O, рівень гри спортсмена – R.

Тоді словесний спосіб подання алгоритму матиме такий вигляд:

1. Ввести оцінку гри гравця O.
2. Проаналізувати, чи є оцінка меншою 4.
3. Якщо так, то рівень гри спортсмена R визнати низьким.
4. Якщо ні, перевірити, чи є оцінка гри меншою 7.
5. Якщо так, то рівень гри спортсмена R визнати середнім.
6. Якщо ні, то перевірити, чи є оцінка гри меншою 10.
7. Якщо так, то рівень гри спортсмена R визнати високим.
8. Інакше рівень гри спортсмена R визнати професійним.
9. Вивести повідомлення про рівень гри спортсмена.

Один зі способів подання блок-схеми представлено на рисунку 3.16.

**Зауваження.** Слід звернути увагу, що розробка блок-схеми – творчий процес. Не має єдиного правильного способу її побудови.

**Циклічний алгоритм** – це алгоритм, у якому деяка частина операцій (**тіло циклу** – послідовність команд) виконується багаторазово. Додатково ко-

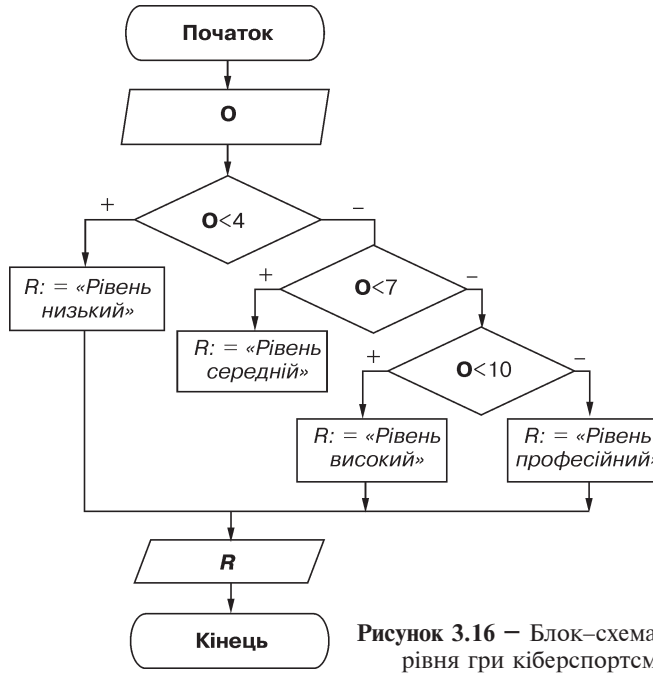


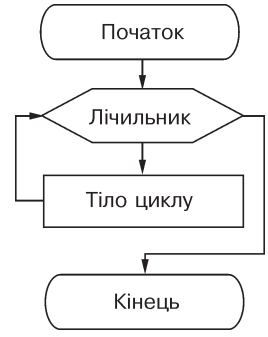

**Рисунок 3.16** – Блок–схема оцінки рівня гри кіберспортсмена

жен цикл має умову, за якою виконання циклічного алгоритму закінчується (рис. 3.17).

**алг** Назва алгоритму (опис аргументів та результатів)
- **арг** Перелік аргументів
- **рез** Перелік результатів

**поч** Опис допоміжних змінних
Введення даних
Дія 1
Дія 2
Вивід результату
**кін**

**Алгоритмічна мова**




**Блок-схема**

**Рисунок 3.17** – Загальний вигляд циклічного алгоритму

Циклічні алгоритми можуть бути з передумовою, з післясловом та з параметром (рис. 3.18).

Щодо графічного подання циклічних алгоритмів, то їх представлено на рисунку 3.19.

Наведемо найпростіший приклад циклічного алгоритму у сфері кіберспорту: поки команда кіберспортсменів із дисципліни Dota 2 проходить попередній етап змагань, в неї є шанси потрапити на найбільший серед інших щорічний кіберспортивний турнір «The International», який проводиться компанією Valve.

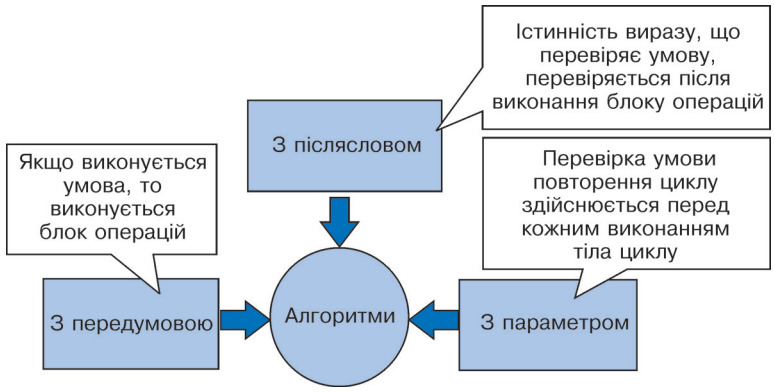



**Рисунок 3.18** – Типи циклічних алгоритмів

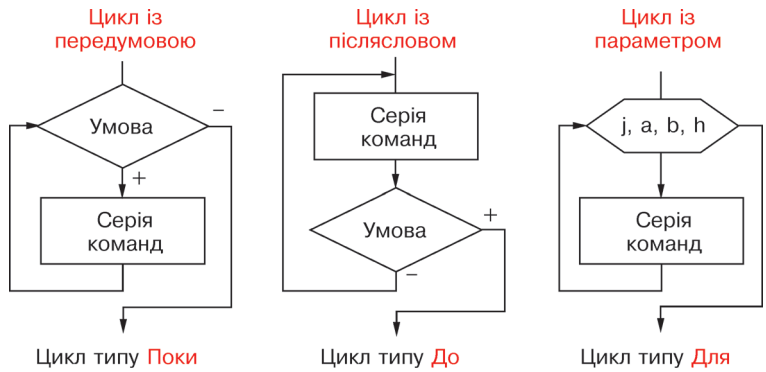



**Рисунок 3.19** – Блок–схеми циклічних алгоритмів

## Правила вибору циклу

1. Якщо відомо заздалегідь число повторень тіла циклу, краще використовувати цикл із передумовою.

2. Якщо заздалегідь невідомо кількість повторень тіла циклу і якщо закінчення циклу залежить від виконання деякої умови, доцільно використовувати цикл із післясловом.

3. Якщо необхідно, щоб цикл завжди виконувався хоча б один раз, то варто використовувати цикл із параметром.

Наступним чином виглядає приклад реалізації циклічного алгоритму на мові С (рис. 3.20).

**Зауваження.** Зверніть увагу, що змінна «і» – параметр циклу, який змінюється всередині циклу за визначеним законом (умовою) і впливає на його закінчення.

**Сортування** – це упорядкування масиву за будь-якою ознакою. Існують різні методи сортування, серед них обмінне (так званий метод пухирця) — найбільш простий. Його доцільно застосовувати для сортування невеликих масивів.

```
#define _CRT_SECURE_NO_WARNINGS// для можливості використовувати scanf
#include <stdio.h>
int main() {
  int k;  // оголошуємо цілу змінну key
  int i = 1;
  int sum = 0; // початкове значення суми рівне 0
  printf("k = ");
  scanf("%d", &k);   // вводимо значення змінної k
  while (i <= k)     // поки і менше або рівне k
  {
    sum = sum + i; // додаємо значення і до суми
    i++;          // збільшуємо і на 1
  }
  printf("sum = %d\n", sum); // виводимо значення суми
  getchar(); getchar();
  return 0;
}
```

**Рисунок 3.20** – Код програми для розрахунку суми чисел від 1 до введеного числа k на мові «С»

Ідея методу полягає у *послідовному порівнянні* пар чисел і їхній *перестановці,* за необхідності. У процесі упорядкування вихідний масив проглядається зліва направо.

**Приклад 1.** Наведемо приклад упорядкування за зростанням методом «пухирця» масив $x_i$ (i = 0...n–1), n = 5, що має значення: $x_0 = 25$, $x_1 = 37$, $x_2 = 0$, $x_3 = 10$, $x_4 = 2$.

*Аналіз задачі.* Для розв'язання задачі необхідно поетапно виконати дії, користуючись такими логічними міркуваннями.

**Перший крок сортування (k = 1):**

Послідовно порівнюються пари сусідніх елементів («25» і «37»), і якщо перший елемент більший за другий, вони міняються місцями, тобто на друге місце, як пухирець, «спливає» більший з двох елементів (у даному випадку елементи залишаються на своїх місцях, елемент, що «спливає», будемо виділяти шрифтом).

**2 5 37 0 10 2**

Потім другий елемент («37»), більший з двох, порівнюється з третім елементом («0»), і на третє місце «спливає» більший з трьох («37»).

**2 5 0 37 10 2**

Далі третій елемент («37») порівнюється з четвертим елементом («10»).

**2 5 0 10 37 2**

Перегляд продовжується до кінця масиву, і найбільший елемент «спливе» та займе останнє місце у масиві.

**2 5 0 10 2 37**

**Другий крок сортування (k = 2)**

Порівнюємо «25» і «0».

**0 2 5 10 2 37**

Продовжуємо порівняння. Унаслідок реалізації другого кроку наступний за значенням найбільший елемент, що «спливе» на передостаннє місце.

**0 10 2 2 5 37**

Аналогічно виконуються кроки 3 і 4. Для масиву з n-елементів за один крок сортування виконується n-1 порівнянь. На кожному новому кроці сортування масиву кількість порівнянь зменшується на одиницю.

Таким чином, окрім блоків введення та виведення елементів масиву, алгоритм сортування методом «пухирця» має два вкладених цикли:

- зовнішній цикл за параметром k – цикл кроків сортування, який керує багаторазовим виконанням внутрішнього циклу;
- внутрішній цикл за параметром i – цикл порівняння елементів та їх перестановки.

Розглянутий приклад наочно ілюструє застосування одного з типових прийомів алгоритмізації – робочої змінної (a), яка необхідна для перестановки місцями двох елементів.

Результат розв'язання задачі наведено на рисунку (рис. 3.21).

**Приклад 2.** Скласти блок-схему потрапляння кіберспортсмена в ТОП-50 гравців.

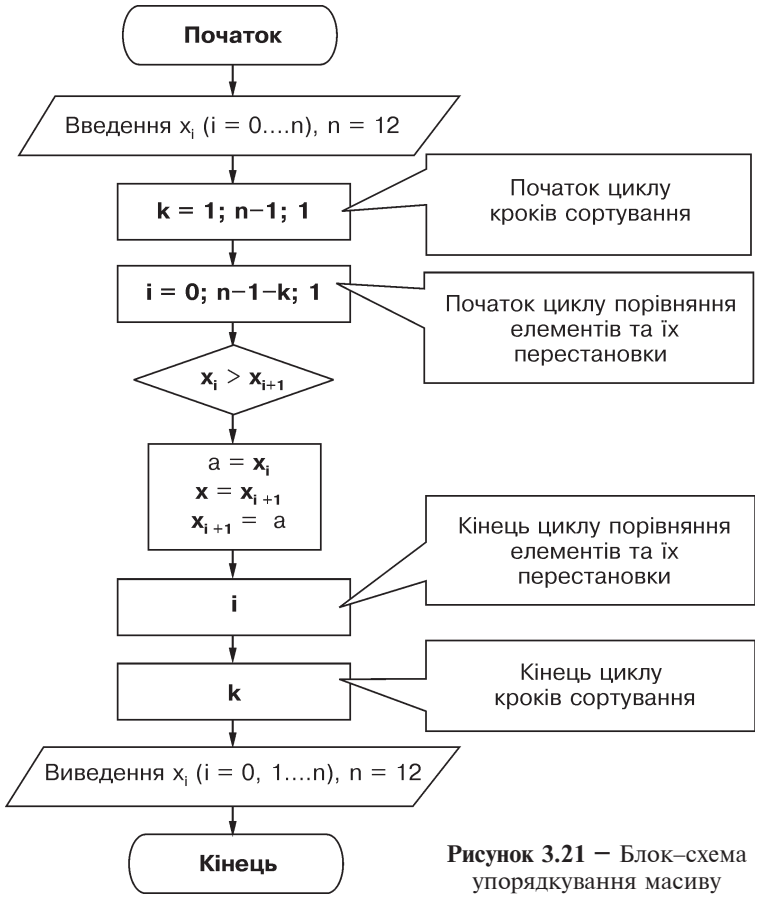



**Рисунок 3.21 –** Блок–схема упорядкування масиву

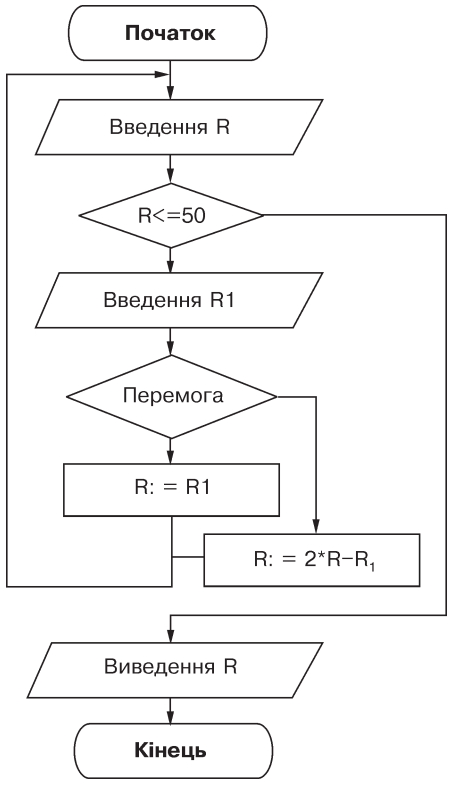



**Рисунок 3.22** – Блок-схема потрапляння гравця в ТОП-50

*Аналіз задачі.* Щоб потрапити в ТОП-50, кіберспортмен повинен підвищувати свій рейтинг, перемагаючи сильних суперників. Для цього необхідно здійснити такі кроки.

1. Перевірити свій поточний рейтинг та місце у рейтинговій таблиці.
2. Якщо він уже у ТОП-50, то закінчити цикл та привітати себе з успіхом.
3. Якщо він ще не в ТОП-50, то знайти суперника, чий рейтинг вищий, ніж у нього.
4. Зіграти з ним матч, використовуючи свої навички та досвід.
5. Якщо він виграв матч, то збільшити свій рейтинг на величину різниці між їхніми рейтингами.
6. Якщо він програв матч, то зменшити свій рейтинг на ту саму величину.
7. Повторити цикл кроку 1.

Результат розв'язання наведено на рисунку 3.22.

## ЗАВДАННЯ ДО ПРАКТИЧНИХ РОБІТ

**Практична робота № 3.** Побудувати блок–схему для визначення, у яких змаганнях може взяти участь кіберспортмен.

Скористатися такими міркуваннями: аматор стане професійним кіберспортсменом, якщо задовольнятиме певним критеріям. Для цього йому потрібно:

- Перевірити рівень гри за шкалою від 1 до 10, де 1 – дуже низький, а 10 – дуже високий.
- Якщо його рівень гри дорівнює 10, він вже є професіоналом і може брати участь у великих турнірах.
- Якщо його рівень гри від 7 до 9, він близький до професійного рівня і може брати участь у кваліфікаціях на турніри або шукати спонсорів та команду.
- Якщо його рівень гри від 4 до 6, то він має середній рівень гри і може продовжувати тренуватися та вдосконалюватись, а також брати участь у аматорських змаганнях.
- Якщо його рівень гри від 1 до 3, то він має низький рівень гри і йому потрібно багато працювати над своїми навичками та знаннями гри, а також вивчати стратегії та тактики найкращих гравців.

**Практична робота № 4.** Розробити алгоритм для побудови рейтингу із 10 кіберспортсменів, що спеціалізуються в Dota 2 (Джерело: https://cq.ru/players), якщо відома сума їхніх призових ($).

| № | Кіберспортсмен | Сума призових, $ |
|---|---|---|
| 1 | electronic | 1685451 |
| 2 | Faker | 1281210 |
| 3 | Zeus | 689829 |
| 4 | Mira | 7567880 |
| 5 | Yatoro | 3929193 |
| 6 | Torontotokyo | 7584830 |
| 7 | s1mple | 1813714 |
| 8 | boombl4 | 1222728 |
| 9 | Collapse | 3930430 |
| 10 | Topson | 11290383 |

*Аналіз задачі.* Як видно з таблиці, для побудови рейтингу гравців потрібно здійснити їхню перестановку залежно від суми призових. При цьому на першому місці буде кіберспортсмен із максимальною сумою призових. Вочевидь, слід скористатися методом «пухирця», який наведено у попередньому прикладі.

## ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ОПРАЦЮВАННЯ

**Завдання 1.** Визначити умови та скласти блок-схему потрапляння команди кіберспортсмів із дисципліни Dota 2 на найбільш престижний турнір The International.

**Завдання 2.** Кіберспортсмен бажає долучитися до команди гравців у Dota 2, де одночасно грають 10 гравців (по 5 гравців у команді). Тренер йому поставив умову: він візьме участь у 10 іграх і принаймні 8 із 10 інших гравців його рекомендуватимуть за результатами гри, то він зарахує його до команди. Скласти блок-схему та алгоритм зарахування кіберспортсмена до команди.

## КОНТРОЛЬНІ ЗАПИТАННЯ ТА ЗАВДАННЯ

***1. Що таке алгоритм? Які існують способи їх представлення?***
***2. У чому полягає завдання алгоритмізації?***
***3. Які головні властивості притаманні алгоритму?***
***4. Які бувають види алгоритмів?***
***5. Чим лінійний алгоритм відрізняється від розгалуженого?***
***6. Дайте визначення розгалуженого алгоритму.***
***7. Назвіть правила побудови алгоритмів за допомогою блок-схем.***
***8. Що являє собою операторний метод подання алгоритму?***
***9. У чому полягають характерні особливості алгоритмічних мов програмування?***
***10. Назвіть види циклічних алгоритмів.***
***11. У чому полягає метод «пухирця»?***
***12. Наведіть приклади алгоритмів у сфері кіберспорту.***

ТЕМА 4

# РОЗВ'ЯЗАННЯ КОНКРЕТНИХ ЗАДАЧ У СФЕРІ КІБЕРСПОРТУ ЗАСОБАМИ ІНФОРМАЦІЙНО–КОМУНІКАТИВНИХ ТЕХНОЛОГІЙ

## ЛЕКЦІЯ 5. ПРИЙНЯТТЯ УПРАВЛІНСЬКИХ РІШЕНЬ У СФЕРІ КІБЕРСПОРТУ

***1. Прикладні задачі оптимізації у сфері кіберспорту.***
***2. Математичне програмування: основні поняття.***
***3. Перспективні напрями застосування методів математичного програмування у кіберспорті:***
***а) задача знаходження підмножини, сума елементів якої рівна заданому числу. Лінійне програмування;***
***б) задача про критичний шлях. Потокове програмування;***
***в) задача комівояжера. Задача про призначення. Цілочислове програмування.***

### ПРИКЛАДНІ ЗАДАЧІ ОПТИМІЗАЦІЇ У СФЕРІ КІБЕРСПОРТУ

У сучасному суспільстві рівень оволодіння фахівцем навичками застосування інформаційних технологій (ІТ) для розв'язання професійно орієнтованих завдань разом з іншими значущими компетентностями визначає його конкурентоспроможність на ринку праці. Серед роботодавців стає дедалі більш затребуваним досвід вирішення практичних завдань засобами інформаційно-комунікаційних технологій (ІКТ), причому це стосується всіх фахівців незалежно від сфери їх професійної діяльності. Підготовка майбутніх фахівців для сфери кіберспорту не є винятком у цих процесах глобальної інформатизації та потребує системного підходу до трансформації змісту освіти.

Фахівцю у сфері кіберспорту доводиться здійснювати пошук оптимальних рішень, тобто фактично розв'язувати такі оптимізаційні задачі, як організація кіберспортивних турнірів, де процес підготовки зводиться до розв'язання задачі пошуку максимального потоку, яку називають задачею про критичний шлях, створення локальної мережі для команди гравців, що також передбачає розв'язання задач потокової оптимізації, зокрема, задачі комівояжера або задачі про найкоротший шлях, розподіл членів команди за типами амплуа або співробітників кіберспортивних клубів за клієнтами для проведення індивідуальних тренувань, що можна розглядати як задачу цілочислового програмування, а саме, задачу про призначення, забезпечення кіберспортивного клубу комп'ютерами, яку доцільно розглядати як типову задачу розподільчого типу, так звану транспортну задачу.

Оволодіння майбутніми фахівцями з кіберспорту методами й прийомами розв'язання професійно орієнтованих задач, спираючись на спрощені моделі, не лише сприяє розвитку в них логічного мислення, а й дозволяє в прикладних дослідженнях глибше зрозуміти суть процесів, що відбуваються, та в практичній діяльності знаходити найбільш вигідні рішення, відхиляючи неефективні. Водночас, попри обчислювальну складність і трудомісткість процесу розв'язання задач на оптимізацію, з розвитком ІТ знаходження оптимального рішення перестає бути прерогативою вузького кола спеціалістів. Використання ІТ знімає обмеження на рівень фундаментальної математичної підготовки студентів та відкриває перспективи для фахівців з фізичної культури і спорту застосовувати потужний математичний апарат для вироблення й обгрунтування рішень методами математичного програмування на основі відпрацьованих алгоритмів.

## МАТЕМАТИЧНЕ ПРОГРАМУВАННЯ: ОСНОВНІ ПОНЯТТЯ

У процесі реалізації управлінських функцій менеджерам кіберспортивної організації та тренерам доводиться приймати багато рішень.

Серед методів прийняття управлінських рішень у різних галузях людської діяльності, де необхідно зробити вибір на користь одного з можливих алгоритму дій, наприклад при вирішенні проблем управління та планування тренувального процесу, розподілу навантаження, планування організації підготовки до змагань використовуються методи математичного програмування. Вирішення управлінських питань шляхом вибору найкращого рішення із альтернативних вивчає математичне програмування, або математична оптимізація.

**Математичне програмування** (МП) – розділ прикладної математики, наука, що вивчає методи оптимізації. Головна мета – знаходження найкращого рішення серед багатьох потенційно можливих відповідно з деяким критерієм ефективності – вибір алгоритму дій.

У 1939 р. академік Л. В. Канторович, досліджуючи деякі задачі економічного змісту, розробив методи чисельного розв'язування екстремальних задач. Із цього часу почався відлік бурхливого розвитку математичного програмування. Серед його розділів найбільш розробленим є лінійне програмування. У 1949 р. американський математик Дж. Данціг опублікував обчислювальний алгоритм для розв'язання задач лінійного програмування – симплекс-метод. Відтоді лінійне програмування і розпочало свій розвиток. Серед розділів математичного програмування виділяють лінійне, нелінійне, стохастичне, динамічне, цілочислове. Основні розділи представлено на рисунку 4.1.

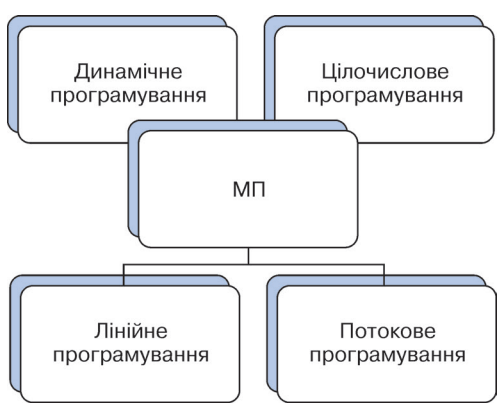



**Рисунок 4.1** – Окремі розділи математичного програмування

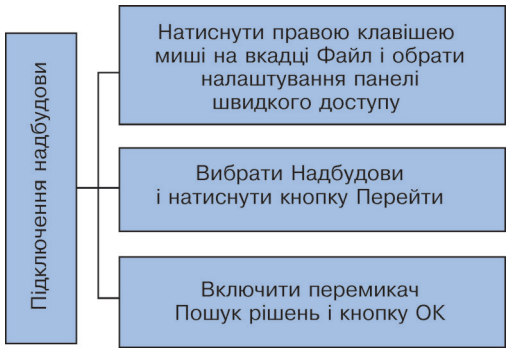



**Рисунок 4.2** – Підключення надбудови Excel Розв'язувач

Модель завдання математичного програмування включає:

- сукупність невідомих величин $x = (x_1, x_2,.., x_n)$, їх називають планом завдання (невідомі величини також можуть бути представлені у вигляді матриці – таблиці чисел $X\{a_{ij}\}$);
- цільову функцію (показник ефективності, критерій оптимальності), яка дозволяє вибирати найкращий варіант із безлічі можливих;
- умову або систему обмежень, що накладаються на невідомі величини.

Сукупність обмежень, записаних у вигляді рівнянь і нерівностей, утворює область **допустимих рішень**.

Допустимий план, при якому цільова функція набуває екстремального значення, називається **оптимальним**.

Знаходити оптимальні результати у завданнях математичного програму-вання можна у програмі MS Excel за допомогою надбудови **Розв'язувач** (рис. 4.2).

## ПЕРСПЕКТИВНІ НАПРЯМИ ЗАСТОСУВАННЯ МЕТОДІВ МАТЕМАТИЧНОГО ПРОГРАМУВАННЯ У КІБЕРСПОРТІ

### Методи пошуку рішення

Для розв'язання задач надбудова пошуку рішення MS Excel пропонує три методи (рис. 4.3).

Алгоритм нелінійної оптимізації Generalized Reduced Gradient (GRG2), розроблений Леоном та Аланом Уореном, використовується для розв'язання нелінійних задач методом зведеного градієнта.

Алгоритми симплексного методу та методу «branch-and-bound» використовуються для вирішення лінійних та завдань із обмеженнями цілочислових, розроблені Джоном Уотсоном і Деном Філстра з компанії Frontline Systems, Inc.

Еволюційний метод також розроблений компанією Frontline Systems, яка є партнером Microsoft з розробки надбудов для MS Excel. Він підходить для ви-

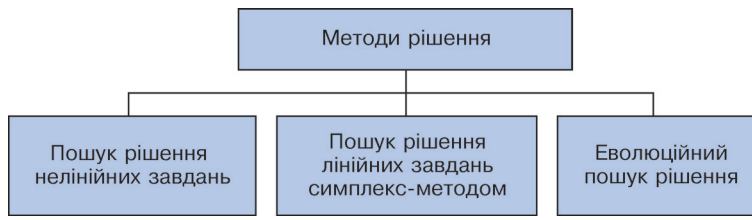



**Рисунок 4.3** – Підключення надбудови Excel Розв'язувач

рішення складних та нелінійних завдань, які не можуть бути вирішені іншими методами. Еволюційний метод розв'язання завдань ґрунтується на принципах еволюції та природного відбору. Він генерує випадкові рішення і потім покращує їх за допомогою мутації, схрещування та відбору.

Під час розв'язання задач ми пропонуємо спочатку скористатися симплекс методом. Якщо розв'язок знайти не вдалося, спробувати знайти рішення методом зведеного градієнта. І, насамкінець, якщо рішення не знайдено, звернутися до еволюційного методу.

Розглянемо приклади розв'язання прикладних задач кіберспорту методами математичного програмування за допомогою надбудови Excel Розв'язувач.

## Задача знаходження підмножини, сума елементів якої рівна заданому числу

**Приклад 1.** *Постановка задачі.* Необхідно заповнити накопичувачі файлами таким чином, щоб порожнє місце, яке залишиться на них після запису всіх файлів, було мінімальним, якщо в наявності є велика кількість файлів.

*Аналіз задачі.*

**Константи** – вихідна інформація. До неї належать розміри файлів, які потрібно записати, та розмір кожного з накопичувачів.

**Змінювані комірки** — діапазон змінних, які потрібно знайти. Це стовпці, де буде отримано відповідь, на який накопичувач слід записати кожен файл. При внесені даних у таблицю Excel вони залишаються порожніми.

**Цільова функція (ЦФ)** – результуючий показник, для якого Excel підбирає найкращі показники. ЦФ задається у вигляді формули в окремій комірці з назвою «Сумарний залишок місця на накопичувачах».

**Напрям оптимізації** – мінімум чи максимум цільової функції. У нашому випадку залишок на накопичувачах після запису файлів має бути мінімальним, отже напрям – мінімум.

**Обмеження** – умови, які необхідно врахувати при оптимізації цільової функції:

- кожен файл записується на один із трьох накопичувачів один раз;
- шукані змінні – бінарні (можуть приймати тільки два значення: Так – 1, Ні – 0);
- залишок на накопичувачах після запису файлів більший нуля.

*Хід роботи.*

1. Ввести початкові дані та обчислити загальний розмір файлів, які потрібно записати.

2. Сформувати необхідні комірки, рядки і стовпці за прикладом, поданим на рисунках 4.4, 4.5.

3. Внести формулу для обчислення потрібної кількості накопичувачів: = ROUNDUP(A19/B21;0).

4. Внести формулу для обчислення комірки «Процент заповнення»: = A19/B22/B23.

5. Сформувати стовпці з наступними іменами «Контроль повторення» та «Розміри файлів на накопичувачах 1, 2, 3».

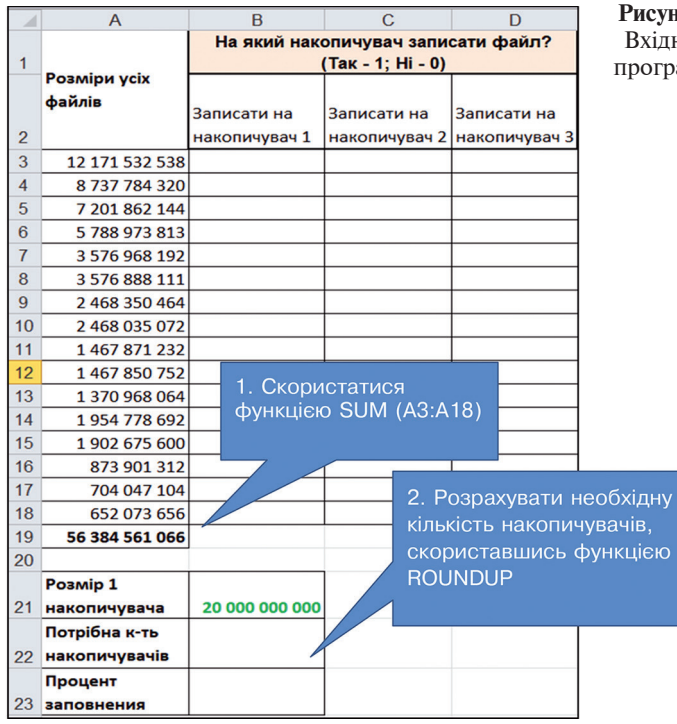



**Рисунок 4.4** – Вхідні дані у програмі Excel

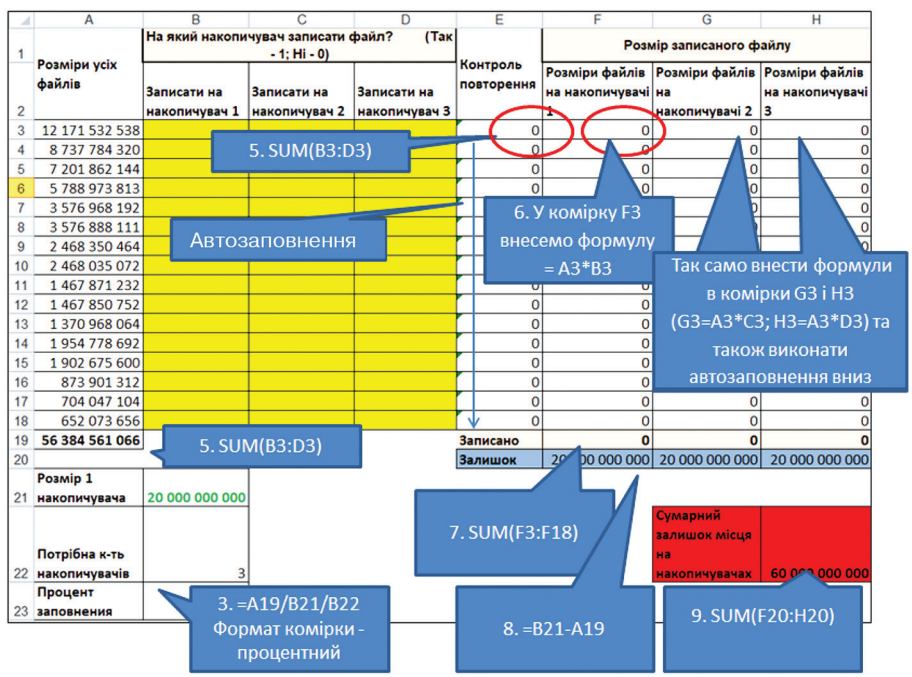



**Рисунок 4.5** – Підготовка даних до розв'язання задачі

6. «Контроль повторень» – файл має бути записаним лише на один накопичувач, тобто сума рядка «Записати на накопичувач...» рівна 1.

7. Розмір файлів на накопичувачах 1, 2, 3 обчислюється як добуток розміру файлу на комірку, де очікується відповідь на питання «Записати файл на накопичувачі».

8. Обчислити ряд «Записано» – розмір записаних файлів на кожному з накопичувачів.

9. Обчислити ряд «Залишок» – вільне місце, що на ньому залишилось (розмір накопичувача відняти розмір записаних файлів).

10. Знайти цільову функцію – місце, що залишиться на накопичувачах – як різницю між загальним розміром накопичувачів і записаними файлами.

## Розв'язання задачі засобами надбудови «Розв'язувач симплекс–методом»

1. Вибрати закладку Дані на панелі інструментів Excel та активізувати надбудову Розв'язувач.

2. У вікні, яке з'явилося, заповнити параметри пошуку рішень (обрати напрям оптимізації; вказати масив невідомих; додати обмеження, відповідно до яких шукані невідомі можуть приймати тільки два значення – бінарні, кожен із файлів можна записати тільки на один із накопичувачів; залишок на накопичувачах після запису більший або рівний нулю) (рис. 4.6).

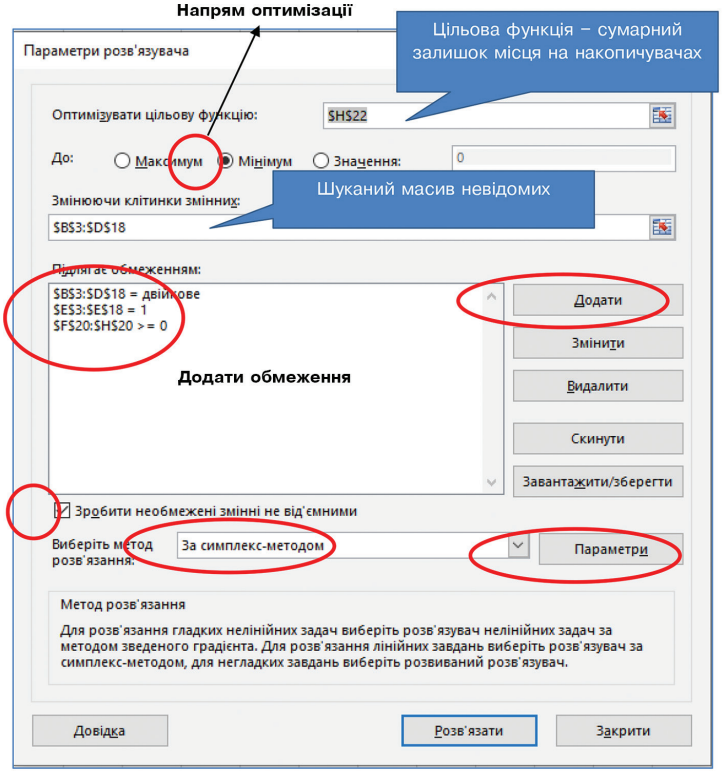



**Рисунок 4.6** – Розв'язання задачі

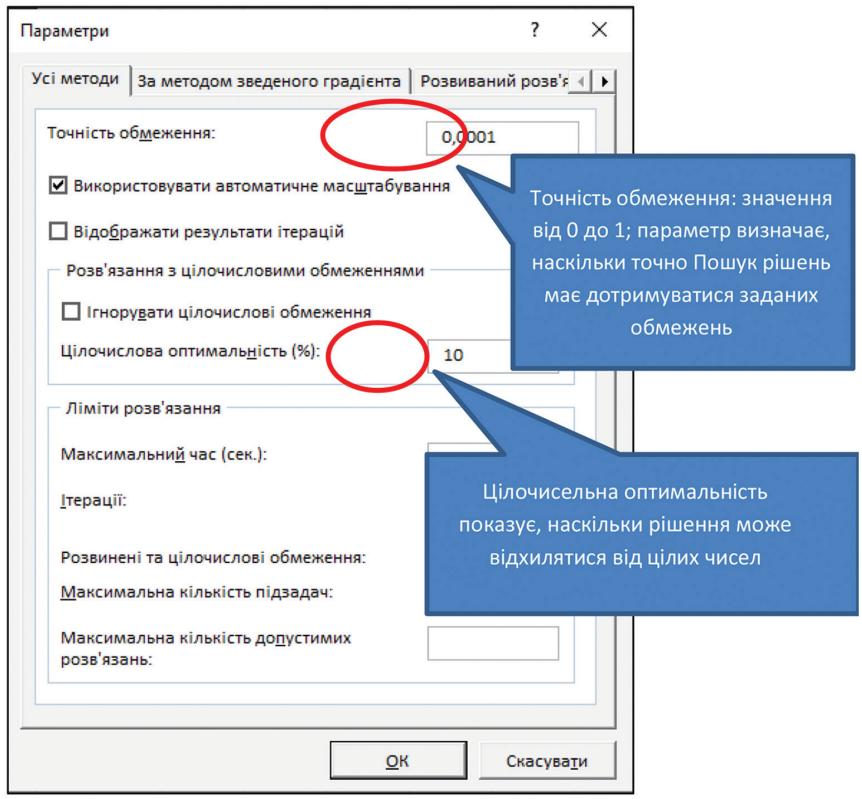



**Рисунок 4.7** – Додаткові параметри

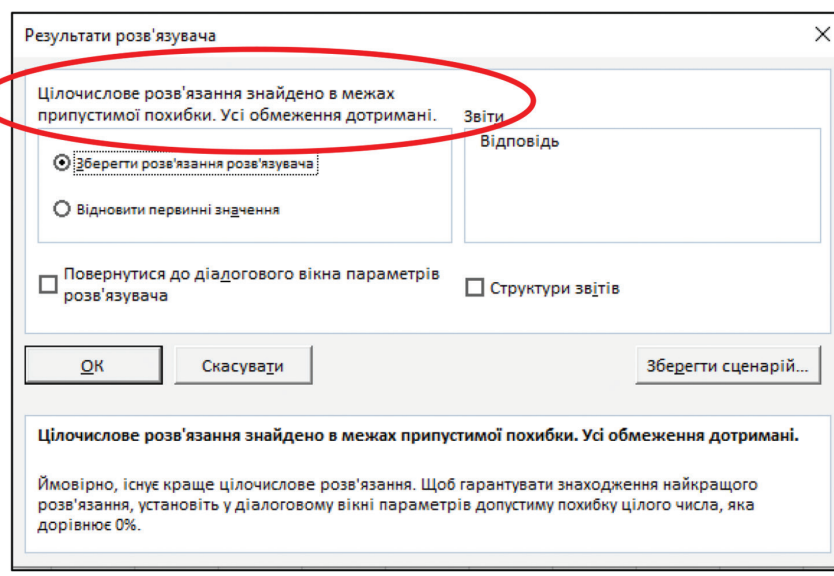



**Рисунок 4.8** – Повідомлення надбудови про знайдене розв’язання

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | Розміри усіх файлів | На який накопичувач записати файл? (Так - 1; Ні - 0) | | | Контроль повторення | Розмір записаного файлу | | |
| 2 | | Записати на накопичувач 1 | Записати на накопичувач 2 | Записати на накопичувач 3 | | Розміри файлів на накопичувачі 1 | Розміри файлів на накопичувачі 2 | Розміри файлів на накопичувачі 3 |
| 3 | 12 171 532 538 | 0 | 1 | 0 | 1 | 0 | 12 171 532 538 | 0 |
| 4 | 8 737 784 320 | 0 | 0 | 1 | 1 | 0 | 0 | 8 737 784 320 |
| 5 | 7 201 862 144 | 0 | 1 | 0 | 1 | 0 | 7 201 862 144 | 0 |
| 6 | 5 788 973 813 | 0 | 0 | 1 | 1 | 0 | 0 | 5 788 973 813 |
| 7 | 3 576 968 192 | 1 | 0 | 0 | 1 | 3 576 968 192 | 0 | 0 |
| 8 | 3 576 888 111 | 1 | 0 | 0 | 1 | 3 576 888 111 | 0 | 0 |
| 9 | 2 468 350 464 | 1 | 0 | 0 | 1 | 2 468 350 464 | 0 | 0 |
| 10 | 2 468 035 072 | 1 | 0 | 0 | 1 | 2 468 035 072 | 0 | 0 |
| 11 | 1 467 871 232 | 1 | 0 | 0 | 1 | 1 467 871 232 | 0 | 0 |
| 12 | 1 467 850 752 | 1 | 0 | 0 | 1 | 1 467 850 752 | 0 | 0 |
| 13 | 1 370 968 064 | 1 | 0 | 0 | 1 | 1 370 968 064 | 0 | 0 |
| 14 | 1 954 778 692 | 1 | 0 | 0 | 1 | 1 954 778 692 | 0 | 0 |
| 15 | 1 902 675 600 | 0 | 0 | 1 | 1 | 0 | 0 | 1 902 675 600 |
| 16 | 873 901 312 | 1 | 0 | 0 | 1 | 873 901 312 | 0 | 0 |
| 17 | 704 047 104 | 1 | 0 | 0 | 1 | 704 047 104 | 0 | 0 |
| 18 | 652 073 656 | 0 | 0 | 1 | 1 | 0 | 0 | 652 073 656 |
| 19 | 56 384 561 066 | | | | Записано | 19 929 658 995 | 19 373 394 682 | 17 081 507 389 |
| 20 | | | | | Залишок | 70 341 005 | 626 605 318 | 2 918 492 611 |
| 21 | Розмір 1 накопичувача | 20 000 000 000 | | | | | | |
| 22 | Потрібна к-ть накопичувачів | 3 | | | | | Сумарний залишок місця на накопичувачах | 3 615 438 934 |
| 23 | Процент заповнення | 93,974% | | | | | | |

**Рисунок 4.9** – Результат розв'язання

3. Налаштувати параметри (рис. 4.7) і натиснути кнопку Знайти рішення (рис. 4.8).

**Зауваження**. При налаштуванні точності слід зважати, що 0 означає, що Пошук рішень буде намагатися знайти рішення, яке точно відповідає заданим обмеженням; 1 означає, що Пошук рішень може відхилятися від заданих обмежень на 100 %.

Відповідно до отриманих результатів можемо прийняти рішення про запис файлів на накопичувачі (рис. 4.9).

***Висновок.*** Приймаємо рішення, на який накопичувач записати файл, аналізуючи, на перетині якого стовпця і рядка з розміром файла відмічено 1. Тоді сумарний залишок на накопичувачах буде мінімальним і становитиме 3615 438 934 байт. При цьому всі файли будуть записаними і накопичувачі заповняться на 93,974 %.

## Задача про критичний шлях. Потокове програмування

**Критичний шлях (КШ)** – це найдовша послідовність операцій від початку проєкту (вузол 1) до його завершення (останній вузол).

Критичний шлях складається із **критичних операцій**, термін виконання яких не можна подовжити, бо тоді не вдасться закінчити проєкт у визначе-

ний в строк. Якщо ж потрібно скоротити час виконання проєкту, то насамперед потрібно скоротити час виконання хоча б однієї роботи на критичному шляху.

Відповідно, мінімальний термін для виконання проєкту дорівнює довжині критичного шляху.

Метод критичного шляху передбачає створення моделі проєкту, що включає такі елементи:

- список усіх операцій, необхідних для виконання проєкту;
- залежності між цими операціями;
- період часу, необхідний для виконання кожної операції (тривалість).

**Приклад 2.** *Постановка задачі.* За вхідними даними необхідно проаналізувати етапи підготовки й проведення турніру з кіберспорту, якщо відома тривалість кожного з етапів (рис. 4.10, 4.11).

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | **Етапи підготовки турніру** | **Початок** | **Кінець** | **Тривалість** | **Дуга (потік)** | **Н-вартість** | **№ вузла** |
| 2 | Попереднє планування | 1 | 2 | 5 | | | 1 |
| 3 | Відбір суддів | 2 | 3 | 7 | | | 2 |
| 4 | Відбір учасників групового етапу (32 команди) | 2 | 4 | 6 | | | 3 |
| 5 | Технічне забезпечення | 2 | 5 | 5 | | | 4 |
| 6 | Інструктаж | 3 | 4 | 2 | | | |
| 7 | Залучення спонсорів | 3 | 6 | 5 | | | |
| 8 | Жеребкування (поділ команд на 4 кошика по 8 команд) | 4 | 5 | 7 | | | 7 |
| 9 | Ігри по системі "кожен із кожним" (1 коло) | 5 | 6 | 8 | | | 8 |
| 10 | Тайм-брейк до першої перемоги однієї з команд | 5 | 7 | 2 | | | 9 |
| 11 | Виступ черлідерів | 6 | 7 | 5 | | | 10 |
| 12 | Жеребкування (поділ команд на 2 кошика по середньому рівню гравців) | 6 | 8 | 5 | | | |
| 13 | Жеребкування (поділ команд на 2 кошика по середньому рівню гравців) | 7 | 8 | 4 | | | |
| 14 | 1/8 півфінала | 7 | 9 | 5 | | | |
| 15 | 1/4 півфінала | 7 | 10 | 5 | | | |
| 16 | 1-й півфінал | 8 | 10 | 5 | | | |
| 17 | Фінал | 9 | 10 | 6 | | | |

Стовпчик невідомих

Наприклад, операція 1/8 півфінала починається з вузла 7, закінчується у вузлі 9 і триває 5 днів

**Рисунок 4.10** – Табличне представлення вхідних даних

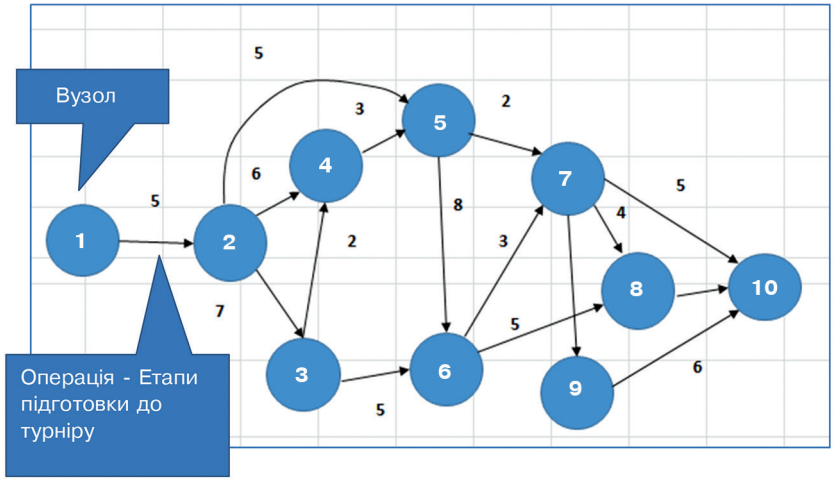



**Рисунок 4.11** – Схематичне представлення вхідних даних

*Аналіз задачі.*

**Константи** – вихідна інформація – початок, кінець і тривалість кожної операції.

**Змінювані комірки** — діапазон змінних – стовпець Дуга (Потік), де буде отримано відповідь, чи є операція критичною.

**Цільова функція (ЦФ)** – тривалість підготовки до турніру.

**Напрям оптимізації** – максимум (мінімальна тривалість підготовки, під час якої всі роботи буде виконано).

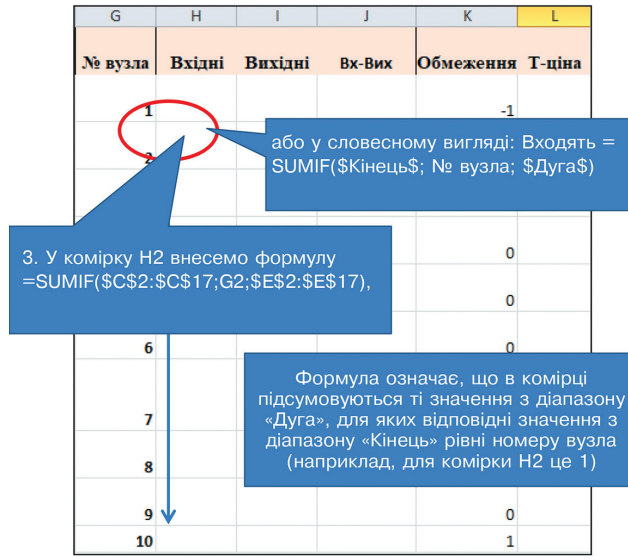



**Рисунок 4.12** – Внесення формул

**Обмеження**. Баланс потоків для кожного з вузлів (рівність суми вхідних та суми вихідних потоків, яка випливає із закону збереження матерії та фізичного стану реального об'єкта, що моделюється вузлом мережі).

*Хід роботи.*

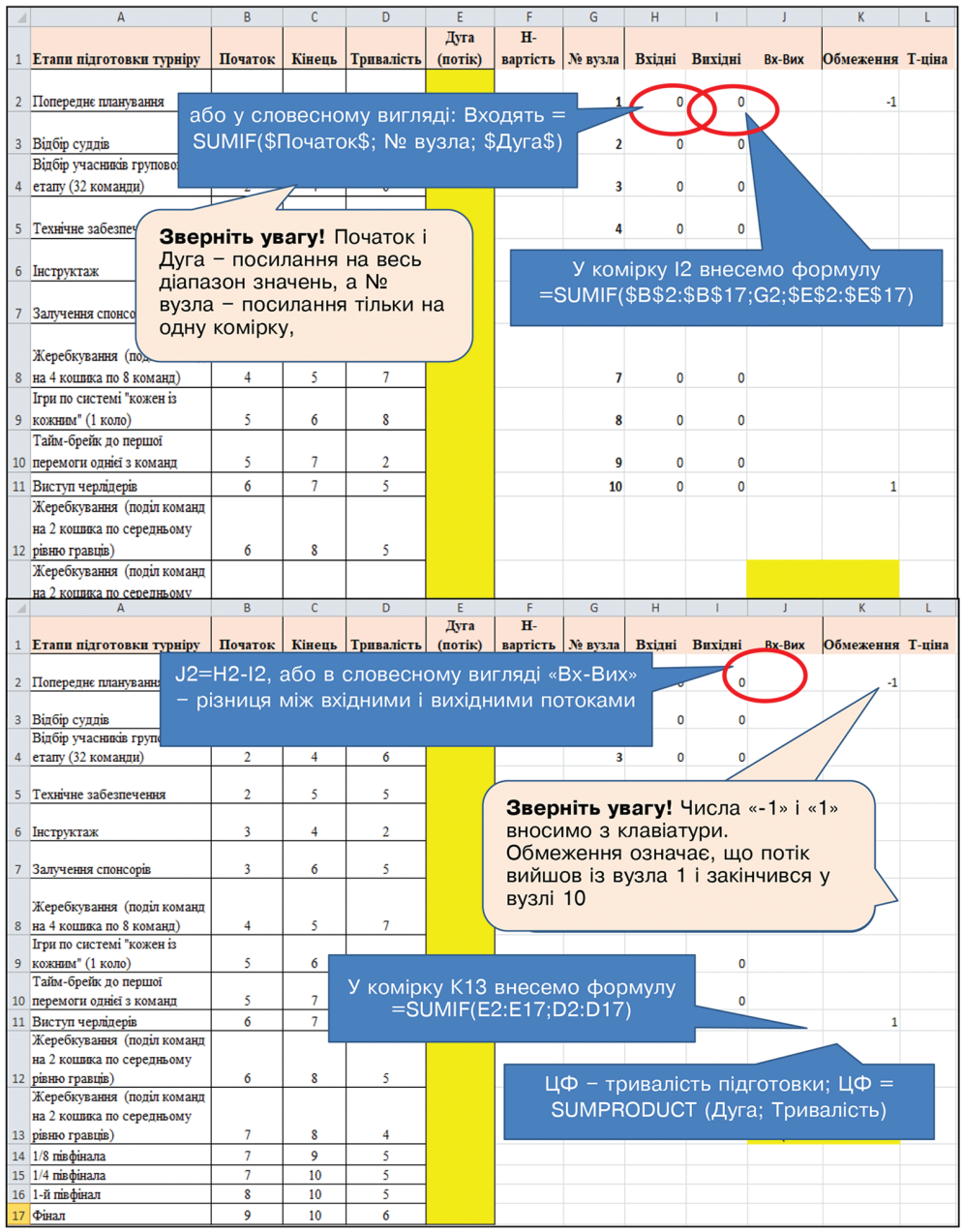
1. Ввести початкові дані.

2. Після стовпця «№ вузла» сформувати стовпці з такими іменами: «Вхідні», «Вихідні», «Вх–Вих» – різниця між вхідними і вихідними потоками, «Обмеження», «Т–ціна» – тіньова ціна, тобто довжина часткового критичного шляху.

3. Внести відповідні формули для розрахунків і виконати автозаповнення (рис. 4.12).

Зауваження 1. Функція SUMIF використовується, якщо необхідно підсумувати значення діапазону, що відповідають зазначеному критерію.

Зауваження 2. Клавішу F4 на клавіатурі натискають, щоб зафіксувати комірку чи діапазон. Тоді з'являються символи $.




| | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Етапи підготовки турніру | Початок | Кінець | Тривалість | Дуга (потік) | Н-вартість | № вузла | Вхідні | Вихідні | Вх-Вих | Обмеження | Т-ціна |
| 2 | Попереднє планування | | | | | | 1 | 0 | 0 | | -1 | |
| 3 | Відбір суддів | | | | | | 2 | 0 | 0 | | | |
| 4 | Відбір учасників групового етапу (32 команди) | | | | | | 3 | 0 | 0 | | | |
| 5 | Технічне забезпеч | | | | | | 4 | 0 | 0 | | | |
| 6 | Інструктаж | | | | | | | | | | | |
| 7 | Залучення спонсо | | | | | | | | | | | |
| 8 | Жеребкування (по на 4 кошика по 8 команд) | 4 | 5 | 7 | | | 7 | 0 | 0 | | | |
| 9 | Ігри по системі "кожен із кожним" (1 коло) | 5 | 6 | 8 | | | 8 | 0 | 0 | | | |
| 10 | Тайм-брейк до першої перемоги однієї з команд | 5 | 7 | 2 | | | 9 | 0 | 0 | | | |
| 11 | Виступ черлідерів | 6 | 7 | 5 | | | 10 | 0 | 0 | | 1 | |
| 12 | Жеребкування (поділ команд на 2 кошика по середньому рівню гравців) | 6 | 8 | 5 | | | | | | | | |
| | Жеребкування (поділ команд на 2 кошика по середньому | | | | | | | | | | | |

| | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Етапи підготовки турніру | Початок | Кінець | Тривалість | Дуга (потік) | Н-вартість | № вузла | Вхідні | Вихідні | Вх-Вих | Обмеження | Т-ціна |
| 2 | Попереднє планування | | | | | | | 0 | | 0 | -1 | |
| 3 | Відбір суддів | | | | | | | 0 | 0 | | | |
| 4 | Відбір учасників груп етапу (32 команди) | 2 | 4 | 6 | | | 3 | 0 | 0 | | | |
| 5 | Технічне забезпечення | 2 | 5 | 5 | | | | | | | | |
| 6 | Інструктаж | 3 | 4 | 2 | | | | | | | | |
| 7 | Залучення спонсорів | 3 | 6 | 5 | | | | | | | | |
| 8 | Жеребкування (поділ команд на 4 кошика по 8 команд) | 4 | 5 | 7 | | | | | | | | |
| 9 | Ігри по системі "кожен із кожним" (1 коло) | 5 | 6 | | | | | | | 0 | | |
| 10 | Тайм-брейк до першої перемоги однієї з команд | 5 | 7 | | | | | | | 0 | | |
| 11 | Виступ черлідерів | 6 | 7 | | | | | | | | 1 | |
| 12 | Жеребкування (поділ команд на 2 кошика по середньому рівню гравців) | 6 | 8 | 5 | | | | | | | | |
| 13 | Жеребкування (поділ команд на 2 кошика по середньому рівню гравців) | 7 | 8 | 4 | | | | | | | | |
| 14 | 1/8 півфінала | 7 | 9 | 5 | | | | | | | | |
| 15 | 1/4 півфінала | 7 | 10 | 5 | | | | | | | | |
| 16 | 1-й півфінал | 8 | 10 | 5 | | | | | | | | |
| 17 | Фінал | 9 | 10 | 6 | | | | | | | | |

**Рисунок 4.13** – Підготовка даних до розв'язання задачі

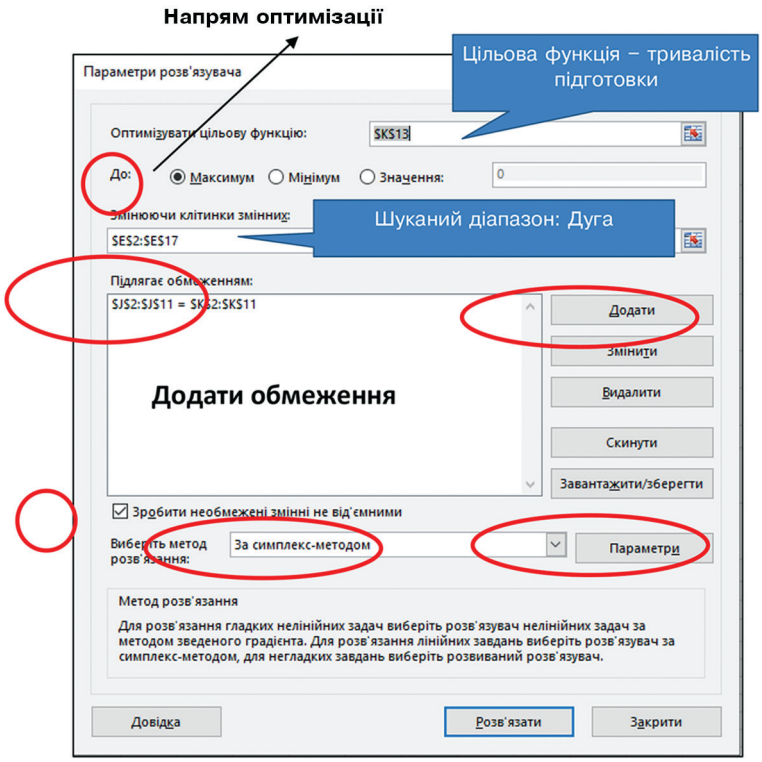



**Рисунок 4.14** – Розв'язання засобами надбудови Розв'язувач

4. Вибрати закладку Дані на панелі інструментів Excel та активізувати надбудову Розв'язувач.

5. У вікні, яке з'явилося, заповнити параметри пошуку рішень і натиснути кнопку Знайти рішення (рис. 4.13, 4.14).

Зауваження 3. Обмеження: стовпчик «Вх–Вих» рівний стовпцю «Обмеження».

## Двоїста задача лінійного програмування

Процес знаходження максимальної тривалості послідовності підготовки до кіберспортивного турніру називають розв'язанням прямої задачі лінійного програмування. У результаті її розв'язання визначають порядок виконання операцій і тривалість усієї підготовки до турніру.

Відповідно, за вихідними даними можна сформулювати **двоїсту** до заданої.

Не заглиблюючись у математичні викладення, розглянемо процес знаходження і інтерпретації розв'язку такої задачі. Зауважимо, що розв'язання двоїстої задачі дозволяє визначити можливості для оптимізації процесу.

Після натиснення Знайти рішення, програма запропонує зберегти звіт розв'язання Розв'язувача (рис. 4.15).

У звіті, що відкривається в окремому аркуші книги Excel, отримаємо розв'язок двоїстої задачі.

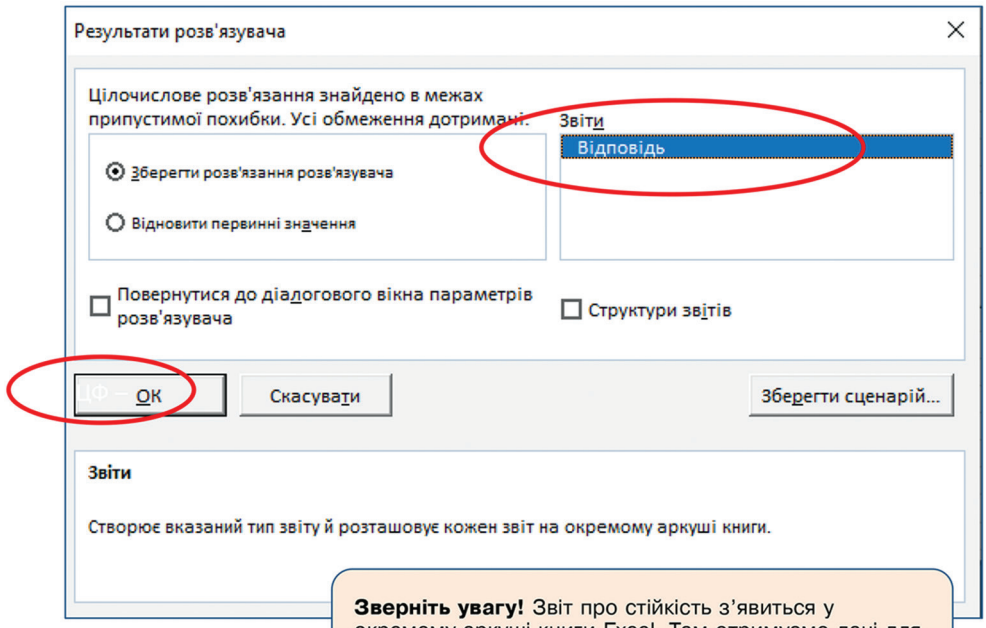



**Рисунок 4.15** – Розв'язання двоїстої задачі

**Зверніть увагу!** Звіт про стійкість з'явиться у окремому аркуші книги Excel. Там отримуємо дані для стовпців «Н-вартість» – нормована вартість, і «Т-ціна».

| | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **Етапи підготовки турніру** | **Початок** | **Кінець** | **Тривалість** | **Дуга (потік)** | **Н-вартість** | **№ вузла** | **Вхідні** | **Вихідні** | **Вх-Вих** | **Обмеження** | **Т-ціна** |
| 2 | Попереднє планування | 1 | 2 | 5 | **1** | 0 | **1** | 0 | 1 | -1 | -1 | 0 |
| 3 | Відбір суддів | 2 | 3 | 7 | **1** | 0 | **2** | 1 | 1 | 0 | 0 | 5 |
| 4 | Відбір учасників групового етапу (32 команди) | 2 | 4 | 6 | 0 | -3 | **3** | 1 | 1 | 0 | 0 | 12 |
| 5 | Технічне забезпечення | 2 | 5 | 5 | 0 | -11 | **4** | 1 | 1 | 0 | 0 | 14 |
| 6 | Інструктаж | 3 | 4 | 2 | **1** | 0 | **5** | 1 | 1 | 0 | 0 | 21 |
| 7 | Залучення спонсорів | 3 | 6 | 5 | 0 | -12 | **6** | 1 | 1 | 0 | 0 | 29 |
| 8 | Жеребкування (поділ команд на 4 кошика по 8 команд) | 4 | 5 | 7 | **1** | 0 | **7** | 1 | 1 | 0 | 0 | 34 |
| 9 | Ігри по системі "кожен із кожним" (1 коло) | 5 | 6 | 8 | **1** | 0 | **8** | 0 | 0 | 0 | 0 | 38 |
| 10 | Тайм-брейк до першої перемоги однієї з команд | 5 | 7 | 2 | 0 | -11 | **9** | 1 | 1 | 0 | 0 | 39 |
| 11 | Виступ черлідерів | 6 | 7 | 5 | **1** | 0 | **10** | 1 | 0 | 1 | 1 | 45 |
| 12 | Жеребкування (поділ команд на 2 кошика по середньому рівню гравців) | 6 | 8 | 5 | 0 | -4 | | | | | | |
| 13 | Жеребкування (поділ команд на 2 кошика по середньому рівню гравців) | 7 | 8 | 4 | 0 | 0 | | | | **ЦФ** | **45** | |
| 14 | 1/8 півфінала | 7 | 9 | 5 | **1** | 0 | | | | | | |
| 15 | 1/4 півфінала | 7 | 10 | 5 | 0 | -6 | | | | | | |
| 16 | 1-й півфінал | 8 | 10 | 5 | 0 | -2 | | | | | | |
| 17 | Фінал | 9 | 10 | 6 | **1** | 0 | | | | | | |

**Рисунок 4.16** – Розв'язання двоїстої задачі

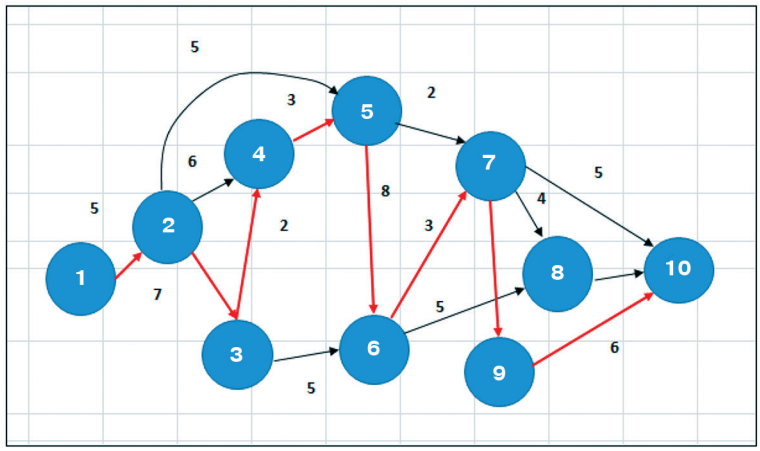



**Рисунок 4.17** – Схематичне представлення порядку підготовки до турніру з кіберспорту

У стовпці **Наведена вартість** (Н-вартість) показано, наскільки зміниться значення в цільовій комірці зі збільшенням значення змінюваної комірки на одну одиницю.

У стовпці **Тіньова ціна** (Т-ціна) міститься інформація про частковий критичний шлях – час виконання операцій від першого до визначеного вузла.

Скопіюємо дані відповідних стовпців і внесемо інформацію у сформовані для цього стопці розрахункової таблиці.

Отримаємо результат розв'язання задачі (рис. 4.16, 4.17).

***Висновок.*** Порядок виконання робіт (1–2,2–3,3–4,4–5,5–6,6–7,7–9,9–10). При цьому термін виконання проєкту становитиме КШ = 45 днів.

Нормовані коефіцієнти некритичних робіт вказують на їхні резерви часу. Наприклад, етап «Технічне забезпечення» можна продовжити на строк до 11 днів і це не вплине на тривалість підготовки до турніру.

Тіньові ціни (Т-ціна) для вузлів визначають часткові КШ від вузла 1 (Початок) до всіх інших вузлів, включаючи вузол 10. Наприклад, з таблиці результатів можна пересвідчитись, що виконання робіт з 1 по 5 вузол включно становить 21 день.

## Задача комівояжера

Задача комівояжера привертає багато уваги дослідників, оскільки до такого класу задач формально зводиться значна кількість прикладних завдань ефективної маршрутизації потоків, у тому числі сировини, інформації, енергоресурсів тощо. Загальна задача комівояжера полягає в побудові в транспортній мережі найкоротшого замкненого маршруту без обмеження в однофазовому відвідуванні кожного пункту.

Слід звернути увагу, що зі збільшенням чисельності об'єктів процес розв'язання задачі зростає, а якщо об'єктів понад 15 – то взагалі її можна розв'язати лише наближеними методами. Причому науковці намагаються знайти найбільш зручні способи її виконання й указують на складність пошуку

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Початкові дані | | | | | | | | | |
| 2 | | Тренер | Капітан команди | Гравець 1 | Гравець 2 | Гравець 3 | Гравець 4 | Гравець 5 | Гравець 6 | Гравець 7 |
| 3 | Тренер | 999 | 9,1 | 15,5 | 10 | 6 | 13 | 16,3 | 17,5 | 23,3 |
| 4 | команди | 9,1 | 999 | 6,7 | 5,1 | 10,3 | 11,7 | 11,2 | 10 | 17,5 |
| 5 | Гравець 1 | 15,5 | 6,7 | 999 | 7,3 | 15,1 | 12,8 | 18,9 | 5 | 13 |
| 6 | Гравець 2 | 10 | 5,1 | 7,3 | 999 | 8 | 6,7 | 6,7 | 7,6 | 17,8 |
| 7 | Гравець 3 | 6 | 10,3 | 15,1 | 8 | 999 | 7,8 | 12,5 | 15,3 | 19,4 |
| 8 | Гравець 4 | 13 | 11,7 | 12,8 | 6,7 | 7,8 | 999 | 6 | 30,5 | 12,1 |
| 9 | Гравець 5 | 16,3 | 11,2 | 18,9 | 6,7 | 12,5 | 6 | 999 | 5 | 24,7 |
| 10 | Гравець 6 | 17,5 | 10 | 5 | 7,6 | 15,3 | 30,5 | 5 | 999 | 8,1 |
| 11 | Гравець 7 | 23,3 | 17,5 | 13 | 17,8 | 19,4 | 12,1 | 24,7 | 8,1 | 999 |

**Рисунок 4.18** – Табличне представлення вхідних даних у програмі Excel

оптимального її розв'язку. Серед різноманіття методів знаходження розв'язку задачі комівояжера виокремлюють повний лексичний перебір, метод найближчого сусіда, метод гілок і меж, метод генетичних алгоритмів, алгоритм мурашиної колонії тощо. Водночас фахівці пропонують розв'язувати задачу, використовуючи рекурентну нейронну мережу або хмарні технології Google Drive.

Ми зосередимося на розв'язанні задачі засобами Розв'язувач надбудови Excel. Зазначимо, що за допомогою стандартного програмного забезпечення, що застосовується в освітньому процесі, існує можливість знаходити замкнений контур для обходу між 9 пунктами.

**Приклад 3.** *Постановка задачі.* Для ефективного здійснення тренувального процесу кіберспортсменів необхідно об'єднати всі їхні комп'ютери в локальну мережу, починаючи й закінчуючи з комп'ютера тренера таким чином, щоб витрати на дріт були мінімальними, причому кожен комп'ютер під'єднується до мережі один раз (рис. 4.18, 4.19).

*Аналіз задачі.*

**Константи** – вихідна інформація – відстань між комп'ютерами кіберспортсменів.

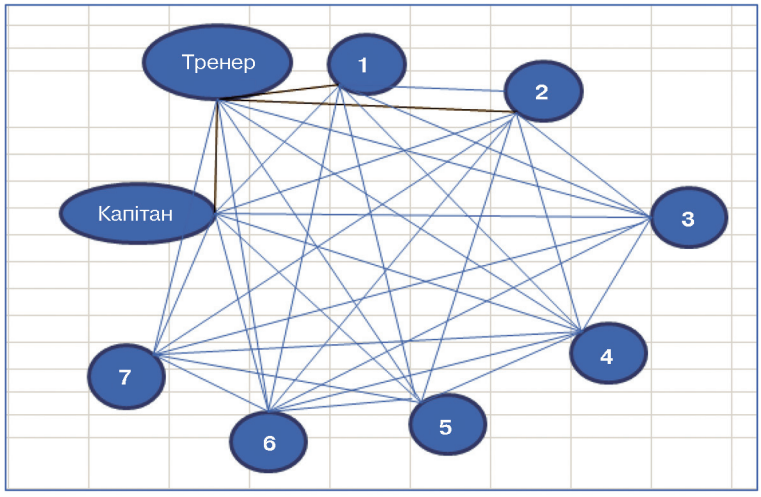



**Рисунок 4.19** – Схематичне представлення усіх можливих варіантів з'єднання комп'ютерів у локальну мережу

**Змінювані комірки** – матриця змінних X. Це таблиця, де на перетині рядків і стовпців отримаємо відповідь на питання, які саме комп'ютери слід з'єднати (1 – Так, 0 – Ні).

**Цільова функція** (ЦФ) – результуючий показник – довжина дроту.

**Напрям оптимізації** – мінімум.

**Обмеження**:

• елементи шуканої матриця X – бінарні (можуть приймати тільки два значення: Так –1, Ні – 0);

• потрібно уникнути замкнутих шляхів, з'єднати 9 комп'ютерів, починаючи і закінчуючи комп'ютером тренера.

*Хід роботи.*

1. Ввести початкові дані.

2. Сформувати матрицю X та підрахувати суми елементів по рядках і стовпцях. Зверніть увагу, спочатку після автозаповнення в рядку Входить і стовпці Виходить стоятимуть винятково нулі (бо наразі матриця порожня) (рис. 4.20).

3. Сформувати матрицю обмежень (рис. 4.21).

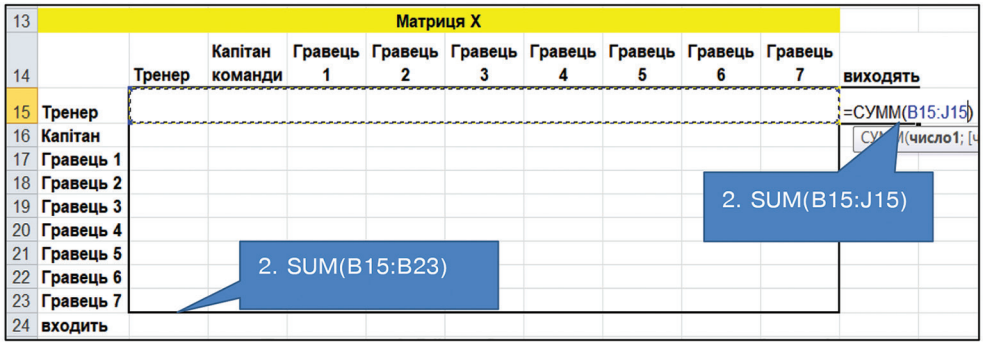



**Рисунок 4.20** – Формування матриці невідомих

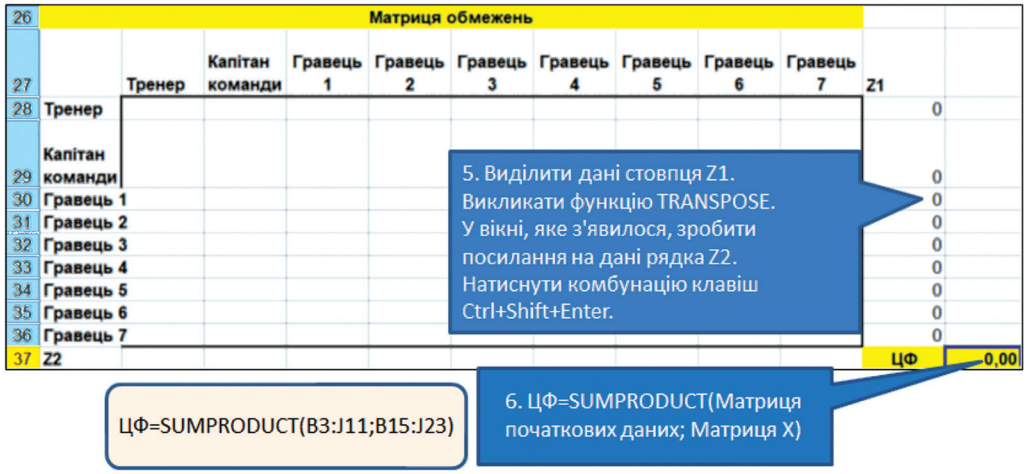



**Рисунок 4.21** – Формування матриці обмежень

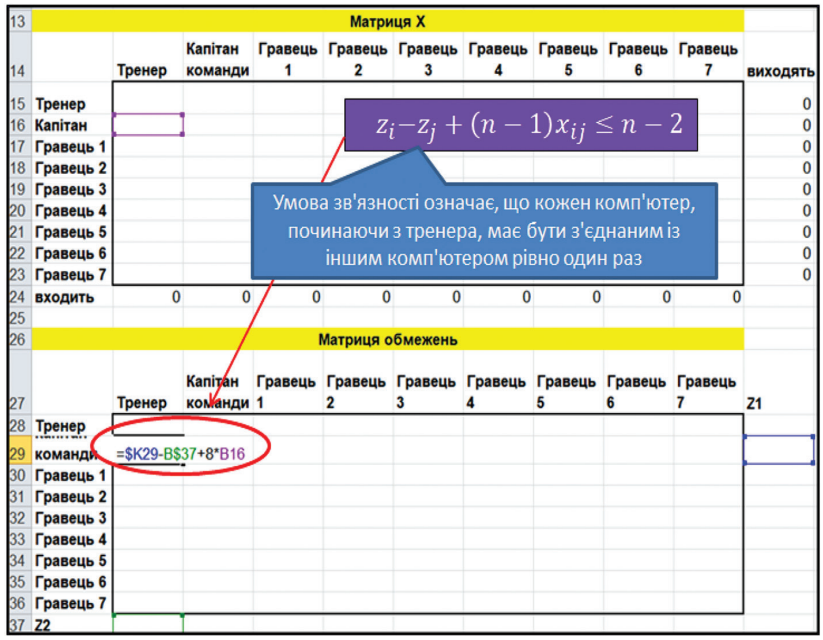



**Рисунок 4.22** – Підготовка даних до розв'язання задачі

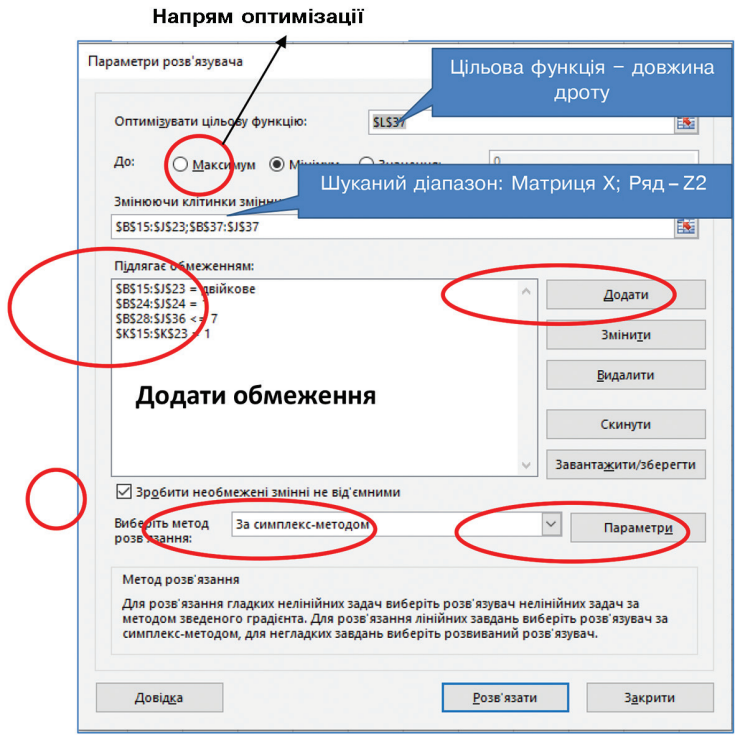



**Рисунок 4.23** – Розв'язання засобами надбудови Розв'язувач

Microsoft Excel 16.0 Звіт про результати
Аркуш: [Комівояжер_3.03.xlsx]Лист3
Звіт створено: 06.02.2024 14:04:01
Результат: Цілочислове розв'язання знайдено в межах припустимої похибки. Усі обмеження дотримані.
Модуль розв'язувача
Модуль: За симплекс-методом
Час розв'язання: 0,64 Секунди.
Ітерації: 20 Підзадачі: 104
Параметри модуля розв'язувача
Максимальний час Без обмежень, Ітерації Без обмежень, Precision 0,000001, Використовувати автоматичне масштабування
Максимальна кількість підзадач: Без обмежень, Максимальна кількість цілочислових розв'язань Без обмежень, Похибка цілого числа 1%, Вважати не від'ємним

Клітинка цільової функції (Мінімум)

| Клітинка | Назва | Вихідне значення | Остаточне значення |
|---|---|---|---|
| $L$37 | ЦФ | 67,40 | 67,40 |

Клітинки змінних

| Клітинка | Назва | Вихідне значення | Остаточне значення | Ціле число |
|---|---|---|---|---|
| $B$15 | Тренер Тренер | 0 | 0 | Двійковий |
| $C$15 | Тренер Капітан команди | 0 | 0 | Двійковий |
| $D$15 | Тренер Гравець 1 | 0 | 0 | Двійковий |
| $E$15 | Тренер Гравець 2 | 0 | 0 | Двійковий |
| $F$15 | Тренер Гравець 3 | 1 | 1 | Двійковий |
| $G$15 | Тренер Гравець 4 | 0 | 0 | Двійковий |
| $H$15 | Тренер Гравець 5 | 0 | 0 | Двійковий |
| $I$15 | Тренер Гравець 6 | 0 | 0 | Двійковий |
| $J$15 | Тренер Гравець 7 | 0 | 0 | Двійковий |
| $B$16 | Капітан команди Тренер | 1 | 1 | Двійковий |
| $C$16 | Капітан команди Капітан команди | 0 | 0 | Двійковий |

**Рисунок 4.24 –** Частина звіту (роз'язання двоїстої задачі)

4. Сформувати діапазон невідомих – ряд Z2.

5. За допомогою функції TRANSPOSE, знайти стовпчик Z1 як матрицю, транспоновану до Z2.

6. Обчислити ЦФ (довжина дроту для локальної мережі).

7. Внести формулу для виконання умови зв'язності, де n – кількість комп'ютерів у локальній мережі (рис. 4.22).

8. Сформувати матрицю обмежень (рис. 4.21).

9. Поширити формулу на всю матрицю обмежень.

10. Вибрати закладку Дані на панелі інструментів Excel та активізувати надбудову Розв'язувач.

11. У вікні, яке з'явилося, заповнити параметри пошуку рішень (Матриця X – бінарна; Матриця обмежень не перевищує 7; ряд «Входить» рівний 1; стовпець «Виходить» рівний 1).

12. Натиснути кнопку Знайти рішення та отримати звіт (рис. 4.23, 4.24, 4.25).

***Висновок.*** Для визначення порядку з'єднання ПК, аналізуємо матрицю X. Починаємо з рядка Тренер. Бачимо, що 1 стоїть на перетині цього рядка зі стовпцем Гравець 3, тобто комп'ютер тренера з'єднуємо з комп'ютером Гравця 3.

Потім переходимо до ряду Гравець 3 і бачимо, що 1 стоїть на перетині зі стовпцем Гравець 4. Отже, прокладаємо дріт до комп'ютера Гравця 4. Продовжуємо аналогічним чином, поки не повернемось до комп'ютера тренера.

Якщо ми таким чином з'єднаємо комп'ютери в локальну мережу, то довжина дроту буде мінімальною і становитиме 67,4 ум. од. (рис. 4.26).

| | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **Початкові дані** | | | | | | | | | | | |
| 2 | | **Тренер** | **Капітан команди** | **Гравець 1** | **Гравець 2** | **Гравець 3** | **Гравець 4** | **Гравець 5** | **Гравець 6** | **Гравець 7** | | |
| 3 | **Тренер** | 999 | 9,1 | 15,5 | 10 | 6 | 13 | 16,3 | 17,5 | 23,3 | | |
| 4 | **Капітан команди** | 9,1 | 999 | 6,7 | 5,1 | 10,3 | 11,7 | 11,2 | 10 | 17,5 | | |
| 5 | **Гравець 1** | 15,5 | 6,7 | 999 | 7,3 | 15,1 | 12,8 | 18,9 | 5 | 13 | | |
| 6 | **Гравець 2** | 10 | 5,1 | 7,3 | 999 | 8 | 6,7 | 6,7 | 7,6 | 17,8 | | |
| 7 | **Гравець 3** | 6 | 10,3 | 15,1 | 8 | 999 | 7,8 | 12,5 | 15,3 | 19,4 | | |
| 8 | **Гравець 4** | 13 | 11,7 | 12,8 | 6,7 | 7,8 | 999 | 6 | 30,5 | 12,1 | | |
| 9 | **Гравець 5** | 16,3 | 11,2 | 18,9 | 6,7 | 12,5 | 6 | 999 | 5 | 24,7 | | |
| 10 | **Гравець 6** | 17,5 | 10 | 5 | 7,6 | 15,3 | 30,5 | 5 | 999 | 8,1 | | |
| 11 | **Гравець 7** | 23,3 | 17,5 | 13 | 17,8 | 19,4 | 12,1 | 24,7 | 8,1 | 999 | | |
| 12 | | | | | | | | | | | | |
| 13 | **Матриця X** | | | | | | | | | | | |
| 14 | | **Тренер** | **Капітан команди** | **Гравець 1** | **Гравець 2** | **Гравець 3** | **Гравець 4** | **Гравець 5** | **Гравець 6** | **Гравець 7** | **виходять** | |
| 15 | **Тренер** | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 1 | |
| 16 | **Капітан команди** | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | |
| 17 | **Гравець 1** | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 1 | |
| 18 | **Гравець 2** | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | |
| 19 | **Гравець 3** | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 1 | |
| 20 | **Гравець 4** | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 1 | |
| 21 | **Гравець 5** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 1 | |
| 22 | **Гравець 6** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 1 | |
| 23 | **Гравець 7** | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | |
| 24 | **входить** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | |
| 25 | | | | | | | | | | | | |
| 26 | **Матриця обмежень** | | | | | | | | | | | |
| 27 | | **Тренер** | **Капітан команди** | **Гравець 1** | **Гравець 2** | **Гравець 3** | **Гравець 4** | **Гравець 5** | **Гравець 6** | **Гравець 7** | **Z1** | |
| 28 | **Тренер** | 0 | 0 | 0 | **0** | 0 | 0 | 0 | 0 | 0 | 8 | |
| 29 | **Капітан команди** | 7 | 0 | 2 | 1 | 7 | 6 | 5 | 4 | 3 | 7 | |
| 30 | **Гравець 1** | -3 | -2 | 0 | 7 | 5 | 4 | 3 | 2 | 1 | 5 | |
| 31 | **Гравець 2** | -2 | 7 | 1 | 0 | 6 | 5 | 4 | 3 | 2 | 6 | |
| 32 | **Гравець 3** | -8 | -7 | -5 | -6 | 0 | 7 | -2 | -3 | -4 | 0 | |
| 33 | **Гравець 4** | -7 | -6 | -4 | -5 | 1 | 0 | 7 | -2 | -3 | 1 | |
| 34 | **Гравець 5** | -6 | -5 | -3 | -4 | 2 | 1 | 0 | 7 | -2 | 2 | |
| 35 | **Гравець 6** | -5 | -4 | -2 | -3 | 3 | 2 | 1 | 0 | 7 | 3 | |
| 36 | **Гравець 7** | -4 | -3 | 7 | -2 | 4 | 3 | 2 | 1 | 0 | 4 | |
| 37 | **Z2** | 8 | 7 | 5 | 6 | 0 | 1 | 2 | 3 | 4 | **ЦФ** | **67,40** |

**Рисунок 4.25** – Результат розв'язання

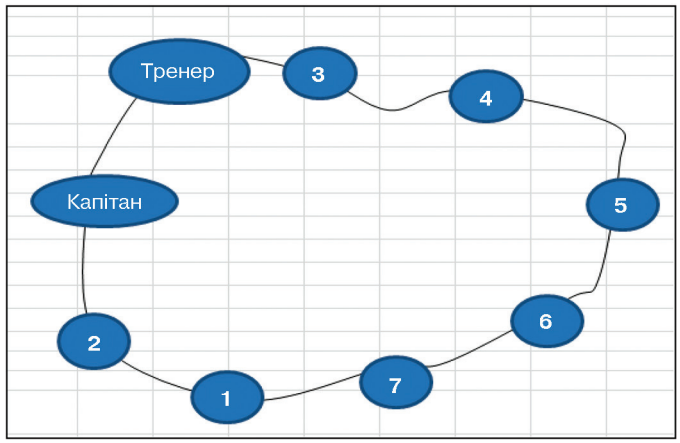



**Рисунок 4.26** – Схематичне представлення порядку з'єднання комп'ютерів

## Задача про призначення

**Приклад 4.** ***Постановка задачі.*** Відома ефективність кожного гравця в команді кіберспортсменів із Dota 2 залежно від амплуа. Розподілити спортсменів таким чином, щоб ефективність команди була максимальною.

Проаналізуємо табличне представлення вхідних даних (рис. 4.27).

*Аналіз задачі.*

**Константи** – вихідна інформація – ефективність кожного кіберспортсмена залежно від амплуа, в якому він грає.

**Змінювані комірки** — матриця змінних X. Це таблиця, де на перетині рядків і стовпців отримаємо відповідь на питання, у якому амплуа виступить кіберспортсмен на турнірі.

**Цільова функція (ЦФ)** – результуючий показник – ефективність команди.

**Напрям оптимізації** – максимум.

**Обмеження:**

- шукана матриця X – бінарна (її елементи можуть приймати тільки два значення: Так –1, Ні – 0);
- кожен із гравців виступить лише в одному амплуа;
- у кожному з амплуа гратиме лише один гравець.

*Хід роботи.*

1. Ввести початкові дані.
2. Сформувати матрицю розподілу кіберспортсменів.
3. Розрахувати суми за рядками і стовпцями (рис. 4.28).
4. Розрахувати цільову функцію – загальна ефективність команди – як добуток матриці ефективності й матриці розподілу гравців (див. рис. 4.28).
5. Вибрати закладку Дані на панелі інструментів Excel та активізувати надбудову Розв'язувач.
6. У вікні, яке з'явилося, заповнити параметри пошуку рішень (ефективність команди – максимальна; ряд «Участь у амплуа» матриці розподілу гравців рівний ряду «Участь у амплуа» матриці ефективності; стовпець «Участь у грі»

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | | Матриця ефективності гравців | | | | | |
| 2 | | Role (Dota 2) | | | | | |
| 3 | Esportsman | Carry | Midlaner | Offlaner | Support | Roamer | Участь у грі |
| 4 | 1 | 3 | 2 | 7 | 6 | 8 | 1 |
| 5 | 2 | 3 | 8 | 2 | 8 | 6 | 1 |
| 6 | 3 | 8 | 7 | 5 | 7 | 3 | 1 |
| 7 | 4 | 6 | 6 | 7 | 6 | 4 | |
| 8 | 5 | 2 | 5 | 3 | 6 | 9 | |
| 9 | Участь у амплуа | 1 | 1 | 1 | 1 | 1 | |

Ряд означає, що в кожному з амплуа гратиме один кіберспортмен

Стовпець означає, що кожен спортсмен гратиме в одному з амплуа

**Рисунок 4.27 –** Табличне представлення вхідних даних

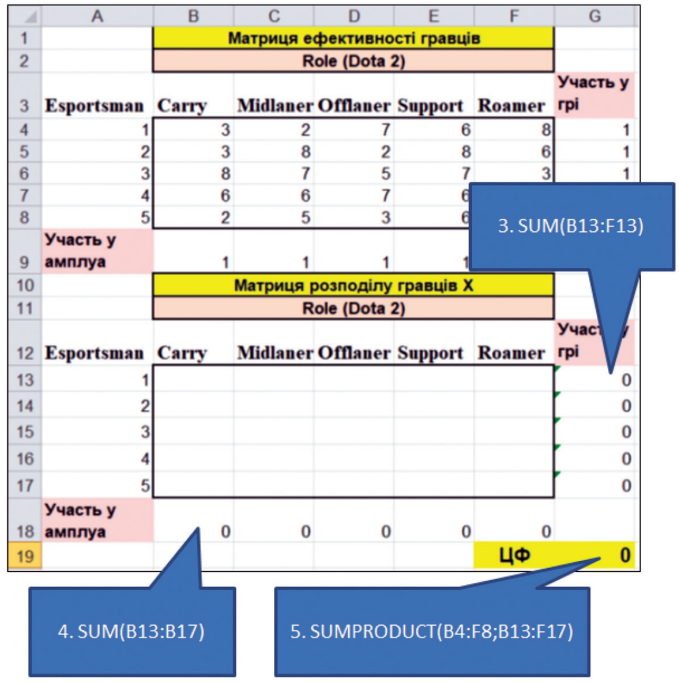



**Рисунок 4.28** – Підготовка даних до розв'язання задачі

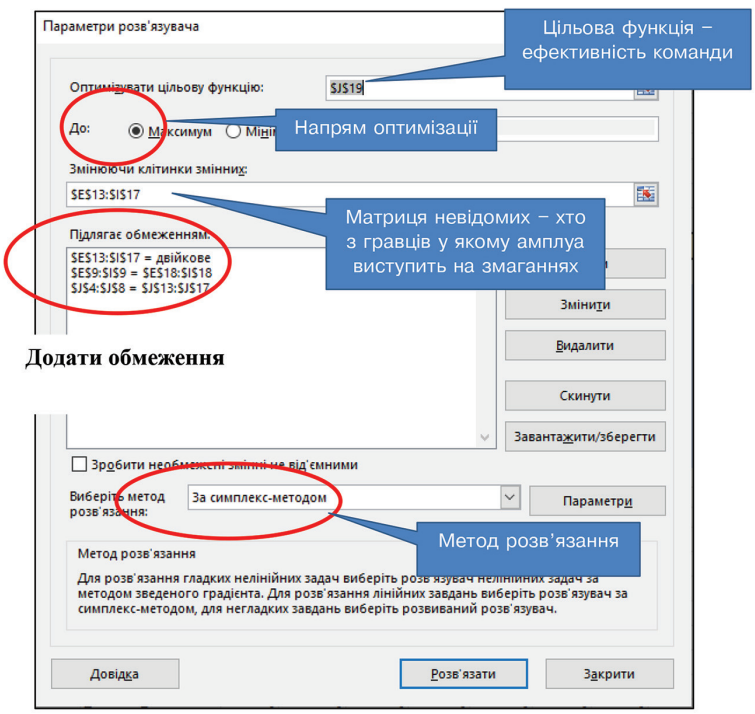



**Рисунок 4.29** – Розв'язання задачі

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | | Матриця ефективності гравців | | | | | |
| 2 | | Role (Dota 2) | | | | | |
| 3 | Esportsman | Carry | Midlaner | Offlaner | Support | Roamer | Участь у грі |
| 4 | 1 | 3 | 2 | 7 | 6 | 8 | 1 |
| 5 | 2 | 3 | 8 | 2 | 8 | 6 | 1 |
| 6 | 3 | 8 | 7 | 5 | 7 | 3 | 1 |
| 7 | 4 | 6 | 6 | 7 | 6 | 4 | 1 |
| 8 | 5 | 2 | 5 | 3 | 6 | 9 | 1 |
| 9 | Участь у амплуа | 1 | 1 | 1 | 1 | 1 | |
| 10 | | Матриця розподілу гравців X | | | | | |
| 11 | | Role (Dota 2) | | | | | |
| 12 | Esportsman | Carry | Midlaner | Offlaner | Support | Roamer | Участь у грі |
| 13 | 1 | 0 | 0 | **1** | 0 | 0 | 1 |
| 14 | 2 | 0 | **1** | 0 | 0 | 0 | 1 |
| 15 | 3 | **1** | 0 | 0 | 0 | 0 | 1 |
| 16 | 4 | 0 | 0 | 0 | **1** | 0 | 1 |
| 17 | 5 | 0 | 0 | 0 | 0 | **1** | 1 |
| 18 | Участь у амплуа | 1 | 1 | 1 | 1 | 1 | |
| 19 | | | | | | ЦФ | 38 |

**Рисунок 4.30** – Результат розв'язання задачі

матриці розподілу гравців рівний стовпцю «Участь у грі» матриці ефективності; матриця розподілу складається з одиниць і нулів – бінарна) (рис. 4.29).

7. Натиснути кнопку Знайти рішення й отримати результат (рис. 4.30).

Слід звернути увагу, що при повторному розв'язанні задачі можуть бути отримані різні розв'язки щодо можливого складу команди за амплуа. Проте в усіх випадках ефективність команди матиме однакове значення – максимальне.

**Висновок.** Приймаємо рішення: на турнірі гравець 1 виступить у амплуа Offlaner, гравець 2 – Midlaner, гравець 3 – Carry, гравець 4 – Support, гравець 5 – Roamer. Тоді ефективність команди буде максимальною і становитиме 38 балів.

**Приклад 5.** ***Постановка задачі.*** Розподілити 6 кіберспортсменів по 4 турнірах, якщо відома ймовірність виграшу кожного гравця на кожному з турнірів, таким чином, щоб ймовірність виграшу була максимальною (рис. 4.31).

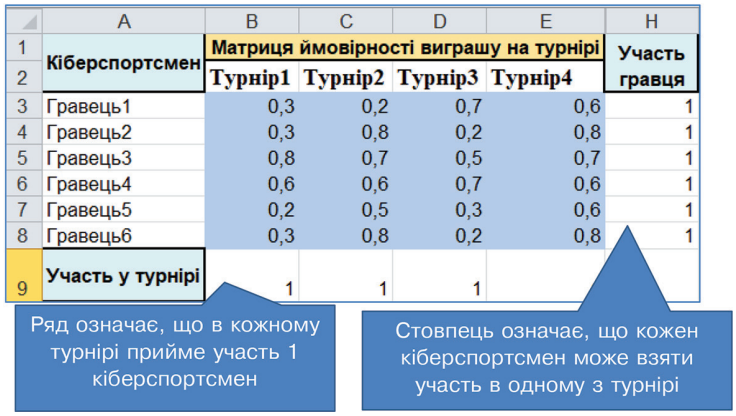
*Аналіз задачі.*

**Константи** – вихідна інформація – ймовірність виграшу кожного гравця на кожному з турнірів.

**Змінювані комірки** — матриця змінних X. Це таблиця, де на перетині рядків і стовпців отримаємо відповідь на питання, у якому турнірі візьме участь той чи інший кіберспортсмен.

**Цільова функція (ЦФ)** – результуючий показник, який визначає вибір кіберспортсменів із максимально можливою ймовірністю перемоги на відповідному турнірі.

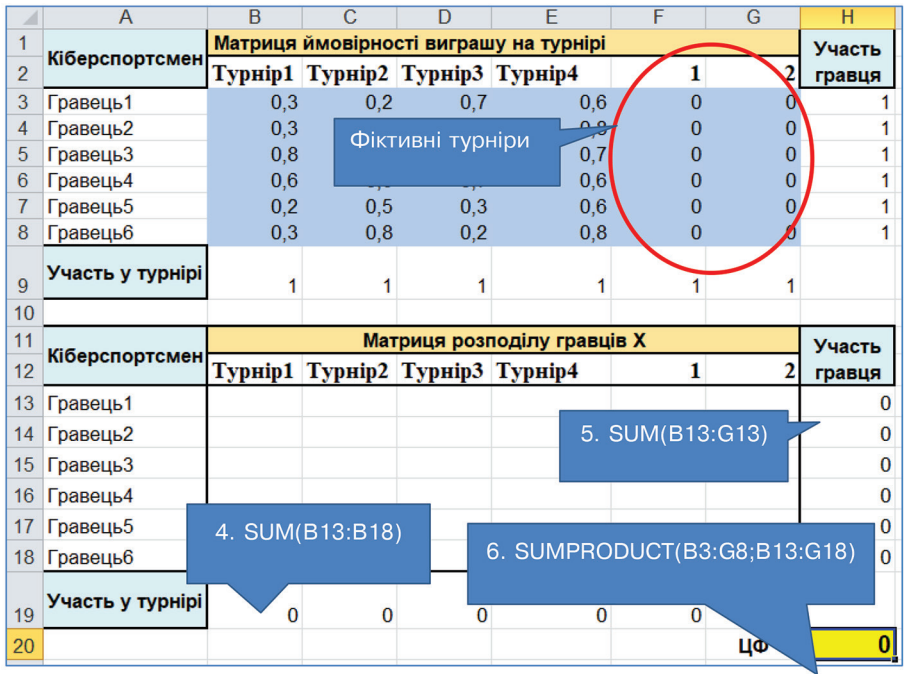
**Напрям оптимізації** – максимум.



| | A | B | C | D | E | H |
|---|---|---|---|---|---|---|
| 1 | Кіберспортсмен | Матриця ймовірності виграшу на турнірі | | | | Участь гравця |
| 2 | | Турнір1 | Турнір2 | Турнір3 | Турнір4 | |
| 3 | Гравець1 | 0,3 | 0,2 | 0,7 | 0,6 | 1 |
| 4 | Гравець2 | 0,3 | 0,8 | 0,2 | 0,8 | 1 |
| 5 | Гравець3 | 0,8 | 0,7 | 0,5 | 0,7 | 1 |
| 6 | Гравець4 | 0,6 | 0,6 | 0,7 | 0,6 | 1 |
| 7 | Гравець5 | 0,2 | 0,5 | 0,3 | 0,6 | 1 |
| 8 | Гравець6 | 0,3 | 0,8 | 0,2 | 0,8 | 1 |
| 9 | Участь у турнірі | 1 | 1 | 1 | | |



**Рисунок 4.31** – Табличне представлення вхідних даних



| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | Кіберспортсмен | Матриця ймовірності виграшу на турнірі | | | | | | Участь гравця |
| 2 | | Турнір1 | Турнір2 | Турнір3 | Турнір4 | 1 | 2 | |
| 3 | Гравець1 | 0,3 | 0,2 | 0,7 | 0,6 | 0 | 0 | 1 |
| 4 | Гравець2 | 0,3 | | | | 0 | 0 | 1 |
| 5 | Гравець3 | 0,8 | | | 0,7 | 0 | 0 | 1 |
| 6 | Гравець4 | 0,6 | | | 0,6 | 0 | 0 | 1 |
| 7 | Гравець5 | 0,2 | 0,5 | 0,3 | 0,6 | 0 | 0 | 1 |
| 8 | Гравець6 | 0,3 | 0,8 | 0,2 | 0,8 | 0 | 0 | 1 |
| 9 | Участь у турнірі | 1 | 1 | 1 | 1 | 1 | 1 | |
| 10 | | | | | | | | |
| 11 | Кіберспортсмен | Матриця розподілу гравців X | | | | | | Участь гравця |
| 12 | | Турнір1 | Турнір2 | Турнір3 | Турнір4 | 1 | 2 | |
| 13 | Гравець1 | | | | | | | 0 |
| 14 | Гравець2 | | | | | | | 0 |
| 15 | Гравець3 | | | | | | | 0 |
| 16 | Гравець4 | | | | | | | 0 |
| 17 | Гравець5 | | | | | | | 0 |
| 18 | Гравець6 | | | | | | | 0 |
| 19 | Участь у турнірі | 0 | 0 | 0 | 0 | 0 | | |
| 20 | | | | | | | ЦФ | 0 |



**Рисунок 4.32** – Підготовка даних до розв'язання задачі

**Обмеження:**

• шукана матриця X – бінарна (її елементи можуть приймати тільки два значення: Так –1, Ні – 0);

• кожен із гравців виступить лише в одному турнірі;

• у кожному турнірі може взяти участь кожен гравець.

*Хід роботи.*

1. Ввести початкові дані.

2. Додати два фіктивні турніри. Прийняти ймовірності виграшу на цих турнірах за нуль.

3. Сформувати матрицю розподілу кіберспортсменів.

4. Розрахувати суми по рядках і стовпцях.

5. Розрахувати цільову функцію – максимально можливі ймовірності виграшу гравців – як добуток матриці ефективності й матриці розподілу гравців (рис. 4.32).

6. Вибрати закладку Дані на панелі інструментів Excel та активізувати надбудову Розв'язувач.

7. У вікні, яке з'явилося, заповнити параметри пошуку рішень (ймовірність виграшу – максимальна; ряд «Участь у турнірі» матриці розподілу гравців рівний ряду «Участь у турнірі» матриці ймовірностей виграшу; стовпець «Участь у грі» матриці розподілу гравців рівний стовпцю «Участь у грі» матриці ймовірностей виграшу; матриця розподілу складається з одиниць і нулів – бінарна).

8. Натиснути кнопку Знайти рішення й отримати попередній результат (рис. 4.33).

***Висновок 1.*** Розрахуємо ймовірність команди кіберспортсменів отримати виграш. Із курсу теорії ймовірностей відомо, що ймовірність P незалежних подій рівна добутку ймовірностей цих подій. Побудуємо матрицю, елементи якої рівні ймовірності виграшу на кожному з турнірів.

Отже, кожен елемент матриці розрахунку ймовірності виграшу рівний добутку відповідних елементів матриці ймовірності виграшу на турнірі та матриці розподілу гравців X.

Тоді ймовірність P рівна добутку 0,8, 0,8, 0,7 і 0,8. Її можна розрахувати за допомогою користувацької формули, перемноживши відповідні елементи.

Для автоматизації процесу розрахунку нам потрібно знайти добуток усіх ненульових елементів матриці розрахунку.

Для цього скористаємося функцією добутку елементів матриці PRODUCT за умови, що ці елементи більші нуля. Отриманий результат матиме такий вигляд (рис. 4.34).

**Зауваження.** Оскільки для обчислення ймовірності **P** використовується формула, яка також є формулою масиву, тому після її введення використовується комбінація клавіш Ctrl+Shift+Enter.

***Висновок 2.*** Якщо розподілити гравців за турнірами запропонованим чином, то ймовірність отримати виграш у команди буде максимальною і становитиме 0,3584 (або 35,84 %).

| Кіберспортсмен | Матриця розподілу гравців X | | | | | | Участь гравця |
|---|---|---|---|---|---|---|---|
| | Турнір1 | Турнір2 | Турнір3 | Турнір4 | 1 | 2 | |
| Гравець1 | 0 | 0 | 0 | 0 | **1** | 0 | 1 |
| Гравець2 | 0 | **1** | 0 | 0 | 0 | 0 | 1 |
| Гравець3 | **1** | 0 | 0 | 0 | 0 | 0 | 1 |
| Гравець4 | 0 | 0 | **1** | 0 | 0 | 0 | 1 |
| Гравець5 | 0 | 0 | 0 | 0 | 0 | **1** | 1 |
| Гравець6 | 0 | 0 | 0 | **1** | 0 | 0 | 1 |
| Участь у турнірі | 1 | 1 | 1 | 1 | 1 | 1 | |
| | | | | | | ЦФ | 3,1 |

**Рисунок 4.33** – Попередній результат розв'язання задачі

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | Кіберспортсмен | Матриця ймовірності виграшу на турнірі | | | | | | Участь гравця |
| 2 | | Турнір1 | Турнір2 | Турнір3 | Турнір4 | 1 | 2 | |
| 3 | Гравець1 | 0,3 | 0,2 | 0,7 | 0,6 | 0 | 0 | 1 |
| 4 | Гравець2 | 0,3 | 0,8 | 0,2 | 0,8 | 0 | 0 | 1 |
| 5 | Гравець3 | 0,8 | 0,7 | 0,5 | 0,7 | 0 | 0 | 1 |
| 6 | Гравець4 | 0,6 | 0,6 | 0,7 | 0,6 | 0 | 0 | 1 |
| 7 | Гравець5 | 0,2 | 0,5 | 0,3 | 0,6 | 0 | 0 | 1 |
| 8 | Гравець6 | 0,3 | 0,8 | 0,2 | 0,8 | 0 | 0 | 1 |
| 9 | Участь у турнірі | 1 | 1 | 1 | 1 | 1 | 1 | |
| 10 | | | | | | | | |
| 11 | Кіберспортсмен | Матриця розподілу гравців X | | | | | | Участь гравця |
| 12 | | Турнір1 | Турнір2 | Турнір3 | Турнір4 | 1 | 2 | |
| 13 | Гравець1 | 0 | 0 | 0 | 0 | **1** | 0 | 1 |
| 14 | Гравець2 | 0 | **1** | 0 | 0 | 0 | 0 | 1 |
| 15 | Гравець3 | **1** | 0 | 0 | 0 | 0 | 0 | 1 |
| 16 | Гравець4 | 0 | 0 | **1** | 0 | 0 | 0 | 1 |
| 17 | Гравець5 | 0 | 0 | 0 | 0 | 0 | **1** | 1 |
| 18 | Гравець6 | 0 | 0 | 0 | **1** | 0 | 0 | 1 |
| 19 | Участь у турнірі | 9. =B3*B13 | | 1 | 1 | 1 | 1 | |
| 20 | | | | | | | ЦФ | 3,1 |
| 21 | Кіберспортсмен | Розрахунок ймовірності виграшу | | | | | | |
| 22 | | Турнір1 | Турнір2 | Турнір3 | Турнір4 | 1 | 2 | |
| 23 | Гравець1 | 0 | 0 | | | | | |
| 24 | Гравець2 | 0 | **0,8** | | | | | |
| 25 | Гравець3 | **0,8** | 0 | | | | | |
| 26 | Гравець4 | 0 | 0 | | | | | |
| 27 | Гравець5 | 0 | 0 | | | | | |
| 28 | Гравець6 | 0 | 0 | 0 | **0,8** | 0 | | |
| 29 | | | | | | | Р | 0,3584 |

10. PRODUCT(IF(B24:G29 > 0;B24:G29) Після введення формули натискаємо комбінацію клавіш Ctrl+Shift+Enter

**Рисунок 4.34** Результат розв'язання задачі

## ЗАВДАННЯ ДО ПРАКТИЧНИХ РОБІТ

**Практична робота № 7.** Для резервного зберігання даних необхідно заповнити один накопичувач файлами відомого розміру. Записати наявні файли таким чином, щоб порожнє місце, яке залишиться на ньому після запису всіх файлів, було мінімальним.

*Аналіз задачі.*

**Константи** – вихідна інформація – розмір файлів; розмір флешки.

**Змінювані комірки** — стовпець «Записати на флешку?» (рис. 4.35).

**Цільова функція (ЦФ)** – результуючий показник – місце, що зали-

| | A |
|---|---|
| 1 | Розміри всіх файлів |
| 2 | 4 171 532 538 |
| 3 | 3 737 784 320 |
| 4 | 3 201 862 144 |
| 5 | 2 576 968 192 |
| 6 | 1 468 350 464 |
| 7 | 1 468 035 072 |
| 8 | 1 467 871 232 |
| 9 | 1 467 850 752 |
| 10 | 1 370 968 064 |
| 11 | 1 302 675 600 |
| 12 | 1 173 901 312 |
| 13 | 704 047 104 |
| 14 | 652 073 656 |

| E | F |
|---|---|
| Розмір флешки | 20 000 000 000 |

**Вхідні дані**

шиться на флешці (різниця між загальним розміром флешки й записаними файлами – формула, введена з клавіатури).

**Напрям оптимізації** – мінімум.

**Обмеження:**

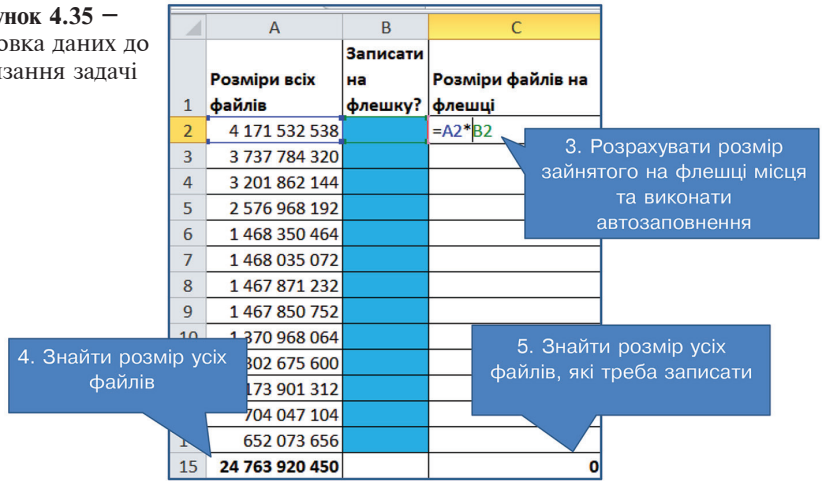
• шукана матриця X – бінарна (елементи стовпця «Записати на флешку?» можуть приймати тільки два значення: Так –1, Ні – 0);

• місце, що лишиться на флешці – більше або рівне 0.

*Підказка.* Для розв'язання задачі необхідно ввести початкові дані, розрахувати й поширити відповідні формули та скористатись надбудовою Excel Розв'язувач (див. рис. 4.35).

Перевірте отриманий результат (рис. 4.36).

**Рисунок 4.35 –** Підготовка даних до розв'язання задачі




| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | **Розміри всіх файлів** | **Записати на флешку?** | **Розміри файлів на флешці** | | **Розмір флешки** | 20 000 000 000 |
| 2 | 4 171 532 538 | **1** | 4 171 532 538 | 1 | | |
| 3 | 3 737 784 320 | **1** | 3 737 784 320 | 2 | | |
| 4 | 3 201 862 144 | **1** | 3 201 862 144 | 3 | | |
| 5 | 2 576 968 192 | **1** | 2 576 968 192 | 4 | | |
| 6 | 1 468 350 464 | **1** | 1 468 350 464 | 5 | | |
| 7 | 1 468 035 072 | **1** | 1 468 035 072 | 6 | | |
| 8 | 1 467 871 232 | **1** | 1 467 871 232 | 7 | | |
| 9 | 1 467 850 752 | **0** | 0 | 8 | | |
| 10 | 1 370 968 064 | **0** | 0 | 9 | | |
| 11 | 1 302 675 600 | **0** | 0 | 10 | | |
| 12 | 1 173 901 312 | **1** | 1 173 901 312 | 11 | | |
| 13 | 704 047 104 | **1** | 704 047 104 | 12 | | |
| 14 | 652 073 656 | **0** | 0 | 13 | | |
| 15 | **24 763 920 450** | | **19 970 352 378** | | | |
| 16 | | | | | | |
| 17 | **Місце, що лишилося на флешці** | | **29 647 622** | | | |

**Рисунок 4.36 –** Результат розв'язання задачі

Вхідні дані

| | Матриця ефективності гравців залежно від амплуа (Dota 2) | | | | | |
|---|---|---|---|---|---|---|
| Кіберспортсмен | Carry | Midlaner | Offlaner | Support | Roamer | Участь у грі |
| Гравець1 | 3 | 2 | 7 | 6 | 8 | 1 |
| Гравець2 | 3 | 8 | 2 | 8 | 6 | 1 |
| Гравець3 | 8 | 7 | 5 | 7 | 3 | 1 |
| Гравець4 | 6 | 6 | 7 | 6 | 4 | 1 |
| Гравець5 | 2 | 5 | 3 | 6 | 9 | 1 |
| Гравець6 | 3 | 8 | 2 | 8 | 4 | 1 |
| Гравець7 | 7 | 7 | 5 | 7 | 3 | 1 |
| Гравець8 | 6 | 2 | 7 | 8 | 4 | 1 |
| Участь у амплуа | 1 | 1 | 1 | 1 | 1 | |

**Рисунок 4.37 –** Табличне представлення вхідних даних

**Рисунок 4.38 –** Обробка вхідних даних

| | Матриця ефективності гравців залежно від амплуа (Dota 2) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Кіберспортсмен | Carry | Midlaner | Offlaner | Support | Roamer | 1 | 2 | 3 | Участь у грі |
| Гравець1 | 3 | 2 | 7 | 6 | 8 | 0 | 0 | 0 | 1 |
| Гравець2 | 3 | 8 | 2 | 8 | 6 | 0 | 0 | 0 | 1 |
| Гравець3 | 8 | 7 | 5 | 7 | 3 | 0 | 0 | 0 | 1 |
| Гравець4 | 6 | 6 | 7 | 6 | 4 | 0 | 0 | 0 | 1 |
| Гравець5 | 2 | 5 | 3 | 6 | 9 | 0 | 0 | 0 | 1 |
| Гравець6 | 3 | 8 | 2 | 8 | 4 | 0 | 0 | 0 | 1 |
| Гравець7 | 7 | 7 | 5 | 7 | 3 | 0 | 0 | 0 | 1 |
| Гравець8 | 6 | 2 | 7 | 8 | 4 | 0 | 0 | 0 | 1 |
| Участь у амплуа | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |

Сформулюйте висновки (які саме файли потрібно записати і скільки на флешці залишиться місця).

**Практична робота № 8.** Для участі в турнірі із дисципліни Dota 2 необхідно відібрати 5 кіберспортсменів із 8 до основного складу команди, якщо відома ефективність кожного гравця у кожному із амплуа, таким чином, щоб ефективність команди була максимальною (рис. 4.37).

*Аналіз задачі.*

**Константи** – вихідна інформація – ефективність гравців; кожен із гравців виступить у одному із амплуа; кожен гравець потрапить або до основного складу команди, або до запасних.

**Змінювані комірки —** матриця невідомих – «Чи потрапить гравець до основного складу і в якому амплуа?».

**Цільова функція (ЦФ)** – результуючий показник – ефективність команди.

**Напрям оптимізації** – максимум.

**Обмеження:**

- шукана матриця X – бінарна (елементи стовпця «Чи потрапить гравець до основного складу і в якому амплуа?» можуть приймати тільки два значення: Так –1, Ні – 0);
- кожен із гравців основної команди виступить лише в одному амплуа і потрапить до основної команди або до запасних.

*Підказка.* У випадках, коли до основного складу команди потрібно відібрати частину гравців, створюється стільки фіктивних амплуа, щоб кількість гравців і кількість амплуа збіглися. У нашому випадку – це 3 (різниця між кількістю гравців і кількістю амплуа становитиме: 8 – 5 = 3). Відповідно, за цими амплуа ефективність гравців становитиме 0 балів (рис. 4.38).

Після цього формується матриця невідомих і розраховуються суми за рядками і стовпцями (рис. 4.39).

Перевірте отриманий результат (рис. 4.40).

**Рисунок 4.39 –** Підготовка даних до розв'язання задачі

| | Матриця невідомих (Чи потрапить гравець до основного складу команди?) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Кіберспортсмен | Carry | Midlaner | Offlaner | Support | Roamer | 1 | 2 | 3 | Участь у грі |
| Гравець1 | | | | | | | | | 0 |
| Гравець2 | | | | | | | | | 0 |
| Гравець3 | | | | | | | | | |
| Гравець4 | | | | | | | | | |
| Гравець5 | | | | | | | | | 0 |
| Гравець6 | | | | | | | | | |
| Гравець7 | | | | | | | | | |
| Гравець8 | | | | | | | | | |
| Участь у амплуа | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| | | | | | | | | ЦФ | 0 |

Суми по рядкам

Суми по стовпцям

Цільова функція – добуток матриці ефективності та матриці невідомих

| | Матриця ефективності гравців залежно від амплуа (Dota 2) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Кіберспортсмен | Carry | Midlaner | Offlaner | Support | Roamer | 1 | 2 | 3 | Участь у грі |
| Гравець1 | 3 | 2 | 7 | 6 | 8 | 0 | 0 | 0 | 1 |
| Гравець2 | 3 | 8 | 2 | 8 | 6 | 0 | 0 | 0 | 1 |
| Гравець3 | 8 | 7 | 5 | 7 | 3 | 0 | 0 | 0 | 1 |
| Гравець4 | 6 | 6 | 7 | 6 | 4 | 0 | 0 | 0 | 1 |
| Гравець5 | 2 | 5 | 3 | 6 | 9 | 0 | 0 | 0 | 1 |
| Гравець6 | 3 | 8 | 2 | 8 | 4 | 0 | 0 | 0 | 1 |
| Гравець7 | 7 | 7 | 5 | 7 | 3 | 0 | 0 | 0 | 1 |
| Гравець8 | 6 | 2 | 7 | 8 | 4 | 0 | 0 | 0 | 1 |
| Участь у амплуа | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |

| | Матриця невідомих (Чи потрапить гравець до основного складу команди?) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Кіберспортсмен | Carry | Midlaner | Offlaner | Support | Roamer | 1 | 2 | 3 | Участь у грі |
| Гравець1 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 1 |
| Гравець2 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Гравець3 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Гравець4 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 1 |
| Гравець5 | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 1 |
| Гравець6 | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 1 |
| Гравець7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 1 |
| Гравець8 | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 1 |
| Участь у амплуа | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| | | | | | | | ЦФ | | 40 |

**Рисунок 4.40 –** Результат розв'язання задачі

**Вхідні дані**

| | Матриця ефективності гравців | | | | | | | | Всього у команді |
|---|---|---|---|---|---|---|---|---|---|
| | Гравці | | | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| Команда1 | 8 | 3 | 5 | 7 | 3 | 8 | 2 | 9 | 4 |
| Команда2 | 8 | 3 | 5 | 7 | 3 | 8 | 2 | 9 | 4 |
| **Участь у команді** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |

**Рисунок 4.41** – Табличне представлення вхідних даних

Сформулюйте висновки (які гравці та в якому амплуа потраплять до складу основної команди й якою буде ефективність цієї команди).

**Практична робота № 9.** Для ефективного здійснення тренувального процесу необхідно розподілити 8 членів команди таким чином, щоб їхня ефективність була подібною (рис. 4.41).

**Зауваження.** Дані за стовпцями дубльовані, оскільки не має відомостей, що ефективність гравця залежить від команди, в якій він гратиме.

*Аналіз задачі.*

**Константи** – вихідна інформація – ефективність гравців; кожен із гравців виступить у одній команді; всього в командах буде по 4 гравці.

**Змінювані комірки** — матриця невідомих – «До якої команди потрапить кіберспортсмен?».

**Цільова функція (ЦФ)** – результуючий показник – відмінності між ефективністю команд (модуль різниці загальної ефективності команд).

**Напрям оптимізації** – мінімум.

**Обмеження:**

- шукана матриця X – бінарна (елементи матриці «До якої команди потрапить кіберспортсмен?» можуть приймати тільки два значення: Так –1, Ні – 0);
- кожен із гравців виступить у одній із команд;
- у кожній команді гратиме 4 кіберспортсмени.

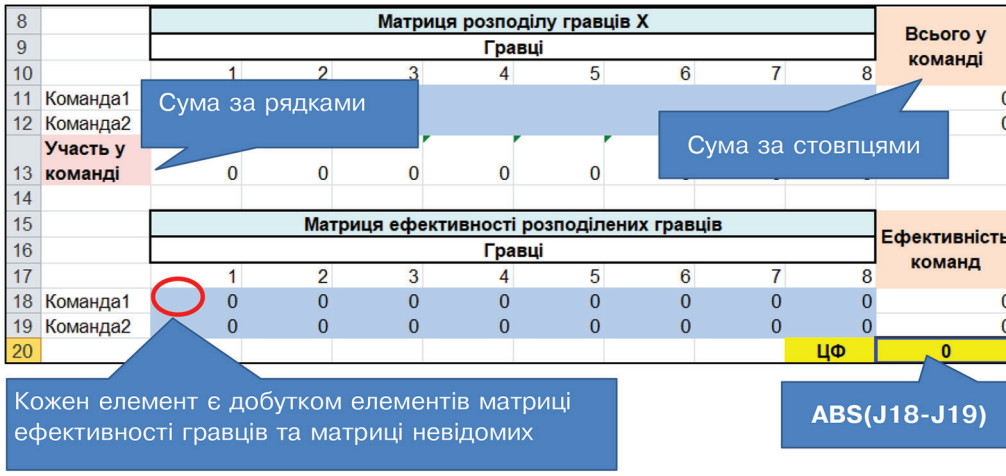



**Рисунок 4.42** – Підготовка даних до розв'язання задачі

*Підказка.* Формуємо матрицю невідомих (Матриця розподілу гравців) і розрахуємо суми за рядками і стовпцями (рис. 4.42).

Формуємо матрицю ефективності розподілених гравців, де кожен її елемент рівний добутку відповідного елемента матриці ефективності гравців та матриці невідомих. Після обчислення першого елемента поширюємо формулу на всю матрицю ефективності розподілених на команди гравців (див. рис. 4.39).

За допомогою функції ABS обчислюємо модуль різниці загальної ефективності команди 1 і загальної ефективності команди 2: ЦФ = ABS(J18–J19).

**Зауваження.** Задачу слід розв'язувати методом УПГ – нелінійний спосіб зведеного градієнта, який використовується для гладких нелінійних задач, тобто за умови нелінійних обмежень.

Перевірте отриманий результат (рис. 4.43).

Сформулюйте висновки (які гравці в яку команду потраплять й якою буде різниця між груповою ефективністю команд).

**Практична робота № 10.** Команда кіберспортсменів готується до турніру. У неї є 4 гравці, кожен з яких може грати у 3 дисципліни. Для кожної дисципліни потрібна певна кількість часу на підготовку: на дисципліну 1 – 50 год, на 2 – 90 год, на 3 – 70 год.

На тренування є 9 днів по 6 год.

Яким чином максимізувати суму годин на підготовку спортсменів, не перевищуючи необхідний час для підготовки до дисциплін, враховуючи, що кожен гравець не може займатися кількома дисциплінами одночасно (рис. 4.44).

| | Матриця ефективності гравців | | | | | | | | Всього у команді |
|---|---|---|---|---|---|---|---|---|---|
| | Гравці | | | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| Команда1 | 8 | 3 | 5 | 7 | 3 | 8 | 2 | 9 | 4 |
| Команда2 | 8 | 3 | 5 | 7 | 3 | 8 | 2 | 9 | 4 |
| **Участь у команді** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |

| | Матриця розподілу гравців X | | | | | | | | Всього у команді |
|---|---|---|---|---|---|---|---|---|---|
| | Гравці | | | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| Команда1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 4 |
| Команда2 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 4 |
| **Участь у команді** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |

| | Матриця результату розподілу | | | | | | | | Ефективність команд |
|---|---|---|---|---|---|---|---|---|---|
| | Гравці | | | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| Команда1 | 8 | 3 | 0 | 0 | 0 | 0 | 2 | 9 | 22 |
| Команда2 | 0 | 0 | 5 | 7 | 3 | 8 | 0 | 0 | 23 |
| | | | | | | | | ЦФ | 1 |

**Рисунок 4.43** – Результат розв'язання задачі

*Аналіз задачі.*

**Константи** – вихідна інформація – загальна кількість часу, доступна для кожного гравця; кількість часу на ефективну підготовку до змагань для кожної дисципліни.

**Змінювані комірки** — матриця невідомих – «Тривалість тренування кожного спортсмена за дисциплінами» (див. рис. 4.44).

**Цільова функція (ЦФ)** – результуючий показник – тривалість підготовки всіх гравців за всіма дисциплінами.

**Напрям оптимізації** – максимум.

**Обмеження:**

- елементи шуканої матриці X – невід'ємні (більші або рівні 0);
- ряд «Підготовка до дисципліни» менший або рівний ряду «Необхідна підготовка до дисципліни»;
- стовпець «Час на тренування» менший або рівний стовпцю «Мінімум необхідного часу».

*Підказка.* Розраховуємо суми за рядками і стовпцями та поширюємо формули на усі дисципліни та усіх гравців.

**Вхідні дані**

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | Матриця X - тривалість тренування кожного спортсмена по дисциплінам | | | Час на тренування |
| 2 | | Дисципліни | | | |
| 3 | Кіберспортсмени | 1 | 2 | 3 | |
| 4 | Гравець 1 | | | | 54 |
| 5 | Гравець 2 | | | | 54 |
| 6 | Гравець 3 | | | | 54 |
| 7 | Гравець 4 | | | | 54 |
| 8 | Необхідна підготовка до дисципліни | 50 | 90 | 70 | |

**Рисунок 4.44** – Табличне представлення вхідних даних

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | | Матриця X - тривалість тренування кожного спортсмена по дисциплінам | | | Час на тренування | Витрачено часу |
| 2 | | Дисципліни | | | | |
| 3 | Кіберспортсмени | 1 | 2 | 3 | | |
| 4 | Гравець 1 | | | | 54 | 0 |
| 5 | Гравець 2 | | | | | 0 |
| 6 | Гравець 3 | | | | | 0 |
| 7 | Гравець 4 | | | | | 0 |
| 8 | Необхідна підготовка до дисципліни | 50 | 90 | 70 | ЦФ | 0 |
| 9 | Тривалість підготовки | 0 | 0 | 0 | | |

Сума часу підготовки кожного гравця за усіма дисциплінами

Сума тривалості підготовки усіх гравців за кожною з дисциплін

СУММ(B4:D7)

**Рисунок 4.45** – Підготовка даних до розв'язання задачі

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | | Матриця X - тривалість тренування кожного спортсмена по дисциплінам | | | Час на тренування | Витрачено часу |
| 2 | | Дисципліни | | | | |
| 3 | Кіберспортсмени | 1 | 2 | 3 | | |
| 4 | Гравець 1 | 0 | 0 | **54** | 54 | 54 |
| 5 | Гравець 2 | 0 | **38** | **16** | 54 | 54 |
| 6 | Гравець 3 | **2** | **52** | 0 | 54 | 54 |
| 7 | Гравець 4 | **48** | 0 | 0 | 54 | 48 |
| 8 | Необхідна підготовка до дисципліни | 50 | 90 | 70 | | 210 |
| 9 | Тривалість підготовки | 50 | 90 | 70 | | |

**Рисунок 4.46** – Результат розв'язання задачі

| | Матриця X - тривалість тренування кожного спортсмена по дисциплінам | | | Час на тренування | Витрачено часу |
|---|---|---|---|---|---|
| | Дисципліни | | | | |
| Кіберспортсмени | 1 | 2 | 3 | | |
| Гравець 1 | 5 | 5 | 44 | 54 | 54 |
| Гравець 2 | 5 | 33 | 16 | 54 | 54 |
| Гравець 3 | 5 | 44 | 5 | 54 | 54 |
| Гравець 4 | 35 | 8 | 5 | 54 | 48 |
| Необхідна підготовка до дисципліни | 50 | 90 | 70 | | 210 |
| Тривалість підготовки | 50 | 90 | 70 | | |

**Рисунок 4.47** – Результат розв'язання задачі з додатковою умовою

Розраховуємо цільову функцію як суму всіх годин, витрачених на тренування (рис. 4.45).

Перевірте отриманий результат (рис. 4.46).

Сформулюйте висновки (скільки годин і в якій саме дисципліні потрібно тренуватися кожному гравцю, щоб оптимально витратити час на тренування і підготуватися до кожної дисципліни).

**Зауваження.** Якщо додаткова умова полягає в тому, щоб на заняття з кожної дисципліни кіберспортсмен витрачав принаймні 5 год, то обмеження «елементи шуканої матриці X – невід'ємні (більші або рівні 0)» замінимо на «елементи шуканої матриці X –більші або рівні 5).

Тоді результат матиме вигляд, наведений на рисунку 4.47.

## ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ОПРАЦЮВАННЯ

**Завдання 1.** Відомі ефективності гравців у команді. Крім того, відомо, що їхня ефективність відрізняється залежно від того, в якій команді гратиме кіберспортсмен. Розподілити 6 членів команди на дві підгрупи таким чином, щоб різниця між їхньою ефективністю була мінімальною.

| | Матриця ефективності гравців | | | | | | Всього у команді |
|---|---|---|---|---|---|---|---|
| | Гравці | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | |
| Команда1 | 8 | 7 | 5 | 7 | 3 | 8 | 3 |
| Команда2 | 6 | 6 | 3 | 6 | 5 | 7 | 3 |
| Участь у команді | 1 | 1 | 1 | 1 | 1 | 1 | |

**Завдання 2.** Команда з 4 кіберспортсменів тренується у 2 дисциплінах. Кожен зі спортсменів має бюджет часу на тренування – 50 год на тиждень. Вони не можуть займатися двома дисциплінами одночасно, а також не можуть витрачати весь час на одну дисципліну, тому що тоді не зможуть підготуватися до іншої. На підготовку до дисципліни 1 потрібно 60 год, на дисципліну 2 – 90 год.

Яким чином максимізувати час підготовки до змагань, не витрачаючи надмірного часу на підготовку до кожної з дисциплін? Як розв'язати попередню задачу за умови, щоб кожен спортсмен принаймні 2 год на тиждень витрачав на підготовку до кожної дисципліни.

## КОНТРОЛЬНІ ЗАПИТАННЯ ТА ЗАВДАННЯ

***1. Що вивчає математичне програмування?***
***2. Які вхідні дані дозволяють сформулювати задачу математичного програмування?***
***3. Які методи розв'язання задач математичного програмування реалізовано в надбудові Excel Розв'язувач?***
***4. Які розділи включає математичне програмування?***
***5. Яке рішення називають допустимим?***
***6. Яке рішення називають оптимальним?***
***7. Який шлях називають критичним?***
***8. Що являє собою цільова функція?***
***9. Що означає «розв'язати двоїсту задачу лінійного програмування»?***
***10. Що таке матриця?***
***11. Чим бінарна матриця відрізняється від інших матриць?***
***12. Назвіть характерні особливості задачі оптимізації.***
***13. Від чого залежить напрям оптимізації?***
***14. Наведіть приклади застосування методів математичного програмування у кіберспорті.***
***15. Яким чином, на вашу думку, фахівець з кіберспорту може використати методи математичного програмування?***
***16. Сформулюйте задачу оптимізації у сфері кіберспорту.***

# ТЕСТИ ДЛЯ ПЕРЕВІРКИ ЗНАНЬ З ДИСЦИПЛІНИ

**1. Що визначає мову програмування?**

1) клас об'єктів даних разом з набором операцій для створення і роботи з ним;

2) сукупність методів виявлення залежності між випадковими величинами;

3) набір лексичних, синтаксичних і семантичних правил;

4) структура послідовності бітів, згрупованих у байти або слова.

**2. Яка з наведених можливостей найкраще описує, як мови програмування використовують для аналітики та оптимізації кіберспортивних змагань?**

1) збір та аналіз даних про гравців і команди;

2) розробка інструментів аналітики та візуалізації даних;

3) автоматизація завдань, пов'язаних з організацією та проведенням кіберспортивних змагань;

4) створення платформ та інструментів для спілкування та співпраці між гравцями, командами та організаторами.

**3. Продовжіть речення: Прикладне програмне забезпечення використовують у кіберспорті в таких напрямах, як:**

1) підготовка фахівців і підвищення кваліфікації; підготовка суддів;

2) спортивне тренування;

3) спортивні змагання;

4) діагностика різних систем організму;

5) науково-дослідна робота.

**4. Що являє собою блок-схема?**

1) візуальне зображення процесу, системи чи алгоритму;

2) схематичне зображення процесу розв'язання задачі за допомогою геометричних фігур;

3) набір інструкцій для розв'язання задачі.

**5. Для чого призначені компілятори?**

1) для перетворення частини програми в машинний код, після чого для іншої частини його використовують знов;

2) для вбудовування додатків;

3) для перетворення тексту програми в машинний код для подальшого використання без нього;

4) для аудіо- та відеотрансляцій.

**6. Що називають алгоритмом?**

1) спосіб подання, зберігання й передачі даних у вигляді фізичних величин, що неперервно змінюються в деякому діапазоні, таких як електричний струм або електрична напруга;

2) систему вказівок, точне й послідовне виконання яких приводить до розв'язання задач;

3) послідовність дискретних (цифрових) значень.

**7. Яку мову програмування вважають мовою четвертого покоління?**

1) Fortran;

2) SQL;

3) ALGOL;

4) Pascal.

**8. Продовжіть речення: Деякий клас об'єктів, даних разом з набором операцій для створення і роботи з ним, – це:**

1) архів;

2) тип даних;

3) ярлик;

4) мова програмування.

**9. Що називають матрицею?**

1) список даних;

2) таблицю, побудовану на основі вхідних даних;

3) множину підвибірок вибіркової сукупності.

**10. Яка з наведених відповідей описує етичну проблему, пов'язану з використанням мов програмування в кіберспорті?**

1) використання алгоритмів для створення упередженості та дискримінації;

2) збір та аналіз особистих даних гравців без їхньої згоди;

3) розробка кіберспортивних ігор, які шкідливі для психічного та фізичного здоров'я;

4) застосування штучного інтелекту для прийняття рішень, які впливають на результат змагань.

**11. Яку математичну модель можна використовувати для оптимізації складу команди з кіберспорту?**

1) модель лінійного програмування;

2) модель нелінійного програмування;

3) модель динамічного програмування;

4) модель цілочислового лінійного програмування.

**12. Яка з наведених відповідей НЕ описує перевагу використання мов програмування в кіберспорті?**

1) підвищення продуктивності та оптимізація роботи;

2) забезпечення прозорості та об'єктивності змагань;

3) створення інструментів для аналітики та тренувань;

4) створення нових кіберспортивних дисциплін;

5) зниження витрат на розробку та підтримку кіберспортивних платформ.

**13. Які способи опису алгоритму вам відомі?**

1) словесно-формульний;

2) графічний;

3) операторний;
4) оперативний;

**14. Як програмісти можуть використати математичне програмування?**

1) для створення моделей реальних проблем, які потім можна вирішити за допомогою комп'ютерних програм;
2) для розробки графічних інтерфейсів користувача (GUI);
3) для створення баз даних;
4) для тестування програмного забезпечення.

**15. Який тип алгоритму використовують для багаторазового виконання одного й того самого блоку коду?**

1) лінійний;
2) регресійний;
3) циклічний;
4) розгалужений;
5) кусково-лінійний.

**16. Які з наведених характеристик є менш пріоритетними для мов програмування п'ятого покоління (5GL)?**

1) використання декларативного програмування;
2) створення алгоритмів на основі природної мови;
3) здатність до самонавчання та самовдосконалення;
4) незалежність від компіляторів та інтерпретаторів;
5) використання паралельних обчислень.

**17. З якою метою використовують надбудову Пошук рішень?**

1) для спрощення процесу обробки даних;
2) для спрощення розрахунків у ході аналізу даних;
3) для знаходження оптимальних рішень з урахуванням змінних і обмежень.

**18. Які з наведених методів оптимізації використовує математичне програмування?**

1) градієнтний спуск;
2) динамічне програмування;
3) симплексний метод;
4) всі згадані.

**19. Яка із наведених мов програмування є алгоритмічною?**

1) пролог;
2) С;
3) Паскаль;
4) SQL;
5) HTML;
6) Бейсик;

**20. Яка з перерахованих характеристик притаманна мові програмування високого рівня?**

1) урахування особливостей конкретних комп'ютерних архітектур;
2) легко переносяться з комп'ютера на комп'ютер;
3) менший розмір програм порівняно з програмами на мові низького рівня;

4) можливість використання комп'ютерів і пристроїв, які мають невеликий обсяг пам'яті.

**21. Як розпочати роботу з надбудовою Пошук рішень?**

1) завантажити надбудову в групі Аналіз;

2) надбудова автоматично завантажується в ході завантаження Microsoft Office;

3) завантажити надбудову в групі Пошук рішень.

**22. Продовжіть речення: Цільова функція – це:**

1) результуючий показник, для якого Excel підбирає найкращі показники;

2) результуючий показник, для якого Excel підбирає максимальний показник;

3) умова, яку необхідно врахувати під час оптимізації цільової функції;

4) змінні, які потрібно знайти.

**23. Що являє собою тіло циклу в циклічному алгоритмі?**

1) послідовність виконуваних команд;

2) послідовність повторюваних команд;

3) умову, яка вимагає перевірки.

**24. Оберіть приклад реальної задачі, яку можна розв'язати за допомогою розгалуженого алгоритму?**

1) перевірка належності гравця до кіберспортивної команди для доступу до платформи для онлайн-змагань;

2) вибір гравця з максимальним рейтингом для участі у турнірі;

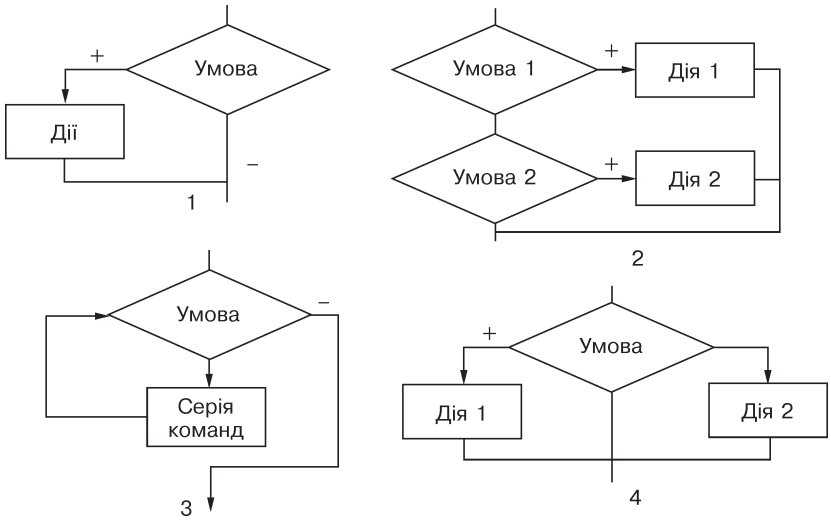
3) визначення амплуа гравця з Dota2 за статистикою героя;

4) обчислення середнього рейтингу команди за рейтингами гравців;

5) розрахунок рейтингу команди протягом року на основі змін у обсязі призових;

6) визначення найкращої команди кіберспорту за результатами одного матчу.

**25. Оберіть із представлених розгалужені алгоритми.**



**26. Назвіть характерні особливості алгоритмічних мов програмування.**

1) жорсткі правила складання речень;

2) як і природна мова, складається зі слів і символів;

3) містить близько 30–40 основних слів;

4) відображають порядок виконання дій за допомогою операторів.

**27. Які з наведених видів алгоритмів є циклічними?**

1) із передумовою;

2) із умовою, залежно від істинності якої виконуються або не виконуються оператори, що входять до складу команди;

3) із післясловом;

4) із параметром.

**28. Чи кожна алгоритмічна мова підходить для програмування?**

1) так;

2) залежно від вирішуваної задачі;

3) ні.

**29. Яка мета застосування методів математичного програмування у сфері кіберспорту?**

1) знаходження найбільш вигідного рішення;

2) вивчення динаміки ефективності гравців під впливом тренувальних навантажень;

3) науково обґрунтоване прийняття управлінських рішень;

4) виявлення і відбір найбільш здібних гравців.

30. **Які із наведених складових визначають задачу оптимізації?**

1) обмеження;

2) умова;

3) цільова функція.

**31. Назвіть тип алгоритму, який використовують для прийняття рішення про те, який блок коду виконати, залежно від значення певної умови?**

1) комбінований;

2) розгалужений;

3) циклічний;

4) лінійний.

**32. Яка з наведених відповідей НЕ описує перевагу використання мов програмування в кіберспорті?**

1) підвищення продуктивності та оптимізація роботи;

2) створення інструментів для аналітики та тренувань;

3) забезпечення чесності та прозорості змагань;

4) створення нових кіберспортивних дисциплін.

# ДОДАТОК

**Таблиця.** Мови програмування

| Рік | Назва | Автор | Характеристика | Приклади використання | Література |
|---|---|---|---|---|---|
| 1949 | Планкалькюль | **Конрад Цузе** | Машина для обчислення формул | Обчислення математичних формул | – |
| 1952 | Машинний код | – | Послідовність машинних кодів, що виконується комп'ютером | Керування роботою комп'ютера | – |
| 1954 | Асемблер | **Джон Мочлі, Джон Басс** | Послідовність мнемонічних кодів, що еквівалентні машинному коду | Керування роботою комп'ютера, написання драйверів пристроїв | – |
| 1956 | FORTRAN | **Джон Басс, Джон К. Річардс, Віктор Б. Вілард** | Перша мову програмування високого рівня | Математичні розрахунки, наукові дослідження | "The Art of Programming" Б. Кернигана та Р. Пайтона |
| 1957 | COBOL | **Грейс Гоппер** | Мова програмування для бізнес-задач | Бізнес-додатки, фінансові системи | "COBOL Programming Language" Г. Гоппер, Дж. Б. Кохена, Р. Л. Сідні |
| 1958 | ALGOL | **Марсель Россі, Кеннет Айверс, Пітер Наур** | Мова програмування для наукових розрахунків | Математичні розрахунки, наукові дослідження | «The Algorithmic Language ALGOL 60» Марселя Россі, Кеннета Айверса, Пітера Наура |
| 1960 | LISP | **Джон Маккарті** | Мова програмування для символьної обробки інформації | Штучний інтелект, обробка природної мови | "The Art of Computer Programming" Д. Кнута |

*Продовження таблиці*

| Рік | Назва | Автор | Характеристика | Приклади використання | Література |
|---|---|---|---|---|---|
| 1964 | BASIC | **Джон К. Гем-мінгс, Говард Гейтс, Річард Гейтс** | Мова програ-мування для початківців | Навчання програ-муванню, прості програми | «Learning Python» Марка Лутца |
| 1967 | Pascal | **Ніколас Вірт** | Мова програ-мування для навчання про-грамуванню | Навчання програ-муванню, струк-турне програму-вання | “The C Prog-ramming Lan-guage” Брайана Кернігана та Денніс Рітчі |
| 1968 | C | **Денніс Рітчі** | Мова програ-мування для загального при-значення | Операційні сис-теми, драйвери пристроїв, висо-копродуктивні програми | “The C Prog-ramming Lan-guage” Брайана Кернігана та Денніс Рітчі |
| 1971 | Simula | **Оле–Йоган Даль, Кристіан Нюгорд, Одд–Івар Фог** | Мова програ-мування для моделювання | Моделювання систем, симуляція процесів | “Object–Oriented Programming: A Modern Approach” Б. Блумстед-та, А. Хомана, Дж. Річардса |
| 1972 | Smalltalk | **Алан Кеймен, Денні Хілл, Боб О’Кейн** | Мова про-грамування для об’єктно-орієнтованого програмування | Об’єктно-орієнтоване про-грамування, інже-нерія програмного забезпечення | “Design Patterns: Elements of Reusable Object-Oriented Software” Е. Гам-ма, Р. Хелм, Р. Джонсон, Дж. Вліссідіс |
| 1975 | C++ | **Б’ярн Страу-струп** | Мова про-грамування для об’єктно-орієнтованого програмування | Об’єктно-орієнтоване програмування, високопродуктив-ні програми | “The C++ Programming Language” Б’єрна Страуструпа |
| 1977 | Modula–2 | **Никлаус Вірт** | Мова програ-мування для структурного програмування | Структурне про-грамування, опе-раційні системи | “The Program-ming Language Modula-2” Нікла-уса Вірта |
| 1983 | Prolog | **Ален Колмеро, Роберт Робінс** | Мова програ-мування для логічного про-грамування | Штучний інте-лект, обробка природної мови | “The Art of Prolog” Л. П. Скотт |
| 1983 | Ada | **Міністерство оборони США** | Мова програ-мування для критичних до безперервності систем | Критичні до безперервності системи, | |

*Продовження таблиці*

| Рік | Назва | Автор | Характеристика | Приклади використання | Література |
|---|---|---|---|---|---|
| 1983 | Perl | **Ларрі Уолл** | Мова програмування для загального призначення | Системне адміністрування, веб-розробка, обробка тексту | “The Perl Programming Language” Ларрі Уолла |
| 1985 | Python | **Гвідо ван Россум** | Мова програмування для загального призначення | Веб-розробка, машинне навчання, наука про дані | “Python in a Nutshell” Чарльза Ніколса |
| 1987 | JavaScript | **Брендан Айх** | Мова програмування для створення інтерактивних вебсторінок | Веб-розробка, створення інтерактивних веб–сторінок | “JavaScript: The Definitive Guide” Дейва Герберта |
| 1988 | Java | **Джеймс Гослінг** | Мова програмування для загального призначення | Веб-розробка, мобільні додатки, хмарна інфраструктура | “The Java Programming Language” Джеймса Гослінг |
| 1990 | PHP | **Расмус Лердорф** | Мова програмування для веброзробки | Веб-розробка, створення статичного та динамічного контенту | “PHP in a Nutshell” Майкла Тісона |
| 1991 | Visual Basic | **Майк Гротц, Девід Міддлтон** | Мова програмування для створення інтерфейсів користувача | Використання з операційними системами Windows | “Visual Basic 6.0 Programmer’s Reference” Майкла Клінтона |
| 1992 | Ruby | **Юкіхіро Мацумото** | Мова програмування для загального призначення | Веб-розробка, машинне навчання, наука про дані | “The Ruby Programming Language” Юкіхіро Мацумото |
| 1993 | Lua | **Робін Годфрід** | Мова сценаріїв для загального призначення | Ігри, віртуальні машини, веб-додатки | “Programming in Lua” Робіна Годфрі |
| 1995 | C# | **Андерс Хейлсберг** | Мова програмування для загального призначення | Веб-розробка, мобільні додатки, хмарна інфраструктура | “The C# Programming Language” Андерса Хейлсберга |
| 1996 | OCaml | **Том Лерой, Франк Баррет** | Мова програмування для загального призначення | Математичні розрахунки, наукові дослідження, машинне навчання | “The OCaml Programming Language” Тома Лероя |
| 1997 | Scala | **Мартін Одерски** | Мова програмування для загального призначення | Об’єктно-орієнтоване програмування, функціональне програмування | “Programming in Scala” Мартіна Одерски |

*Продовження таблиці*

| Рік | Назва | Автор | Характеристика | Приклади використання | Література |
|---|---|---|---|---|---|
| 1998 | Erlang | **Джозеф Армстронг** | Мова програмування для розподілених систем | Розподілені системи, телекомунікації | «Erlang Programming» Джозефа Армстронга |
| 1999 | Haskell | **Дональда Пікардта** | Мова програмування для функціонального програмування | Функціональне програмування, машинне навчання | “Learn You a Haskell for Great Good!” Саймона Петерсона |
| 2000 | Clojure | **Річард Хиршберг** | Мова програмування для функціонального програмування | Функціональне програмування, машинне навчання | «Clojure Programming» Річарда Хиршберга |
| 2001 | Groovy | **Джейсон Сміт** | Мова програмування для загального призначення | Веб-розробка, мобільні додатки, хмарна інфраструктура | “Groovy in Action” Джейсона Сміта |
| 2003 | F# | **Microsoft** | Мова програмування для загального призначення | Об’єктно-орієнтоване програмування, функціональне програмування | “F# for Fun and Profit” Ендрю Джойса |
| 2004 | Rust | **Крістофер Томпсон** | Мова програмування для загального призначення | Безпечне програмування, високопродуктивні програми | “The Rust Programming Language” Крістофера Томпсона |
| 2005 | Go | **Google** | Мова програмування для загального призначення | Інфраструктурні проекти, розподілені системи | “The Go Programming Language” Робіна Хік |

# ЛІТЕРАТУРА


1. Бишевець Н. Г. Динаміка рівня знань з вищої математики студентів ВНЗ в залежності від напрямку навчання / Н. Г. Бишевець // Results of International scientific–practical web-congress of pedagogues and psychologists «Be smart!» the 17-18th of February– 2015. – Т. 1. – С. 60–69.

2. Бишевець Н. Г. Динаміка рівня математичної підготовки студентів ВНЗ під впливом авторської технології навчання математичних дисциплін / Н. Г. Бишевець // Наук. часоп. НПУ ім. М. П. Драгоманова. – 2015. – № 15. – С. 14–21.

3. Бишевець Н. Г. Критерії ефективності технології навчання математичних дисциплін / Н. Г. Бишевець // Мат. міжнар. наук.-практ. конф. – К.: ГО «Київська наукова організація педагогіки та психології». – 2015. – С. 27–30.

4. Бишевець Н. Г. Математична підготовка студентів вищих навчальних закладів: зб. тез міжнар. наук.-практ. конф. / Н. Г. Бишевець. – Л.: ГО «Львівська педагогічна спільнота», 2015. – С. 69–71.

5. Бишевець Н. Г. Мотивація студентів ВНЗ до вивчення математичних дисциплін в залежності від напрямку навчання / Н. Г. Бишевець // Scientific and practical edition: Austria. – 2015. – Т. 2. – С. 17–18.

6. Бишевець Н. Г. Оцінка рівня математичної підготовки студентів вищих навчальних закладів / Н. Г. Бишевець // Мат. міжнар. наук.-практ. конф. – 2015. – С. 13–17.

7. Бишевець Н. Г. Технологія навчання математичних дисциплін студентів вищих навчальних закладів / Н. Г. Бишевець // Укр. психол.-пед. наук. зб. – 2015. – № 4 (04). – С. 34–37.

8. Бишевець Н. Г. Методика застосування електронного навчально-методичного комплексу на практичних заняттях з математичного програмування // Наук. часоп. Нац. пед. ун-ту ім. М. П. Драгоманова. Сер. 3: Фізика і математика у вищій і середній школі. – 2016. – Вип. 17. – С. 43–48.

9. Бишевець Н. Г. Підготовка студентів вищих навчальних закладів до виконання інженерних розрахунків / Н. Г. Бишевець // Укр. психол.-педагог. наук. зб. – 2016. – С. 154–166.

10. Бишевець Н. Г. Досвід застосування сучасних засобів навчання на практичних заняттях з теорії ймовірностей / Н. Г. Бишевець // Інформаційні технології в освіті. – 2017. – № 2 (31). – С. 95–108.

11. Бишевець Н. Г. Мотивація до навчальної діяльності майбутніх фахівців із фізичної культури та спорту в процесі оволодіння методами комп'ютерного моделювання / Н. Г. Бишевець, Н. М. Гончарова // Наук.-метод. основи використання інформ. техн. в галузі фіз. культури і спорту: зб. наук. пр. – Х.: ХДАФК. – 2020. – № 4. – С. 15–9.

12. Бишевець Н. Формування навичок застосування інформаційних технологій у майбутніх фахівців із фізичної культури та спорту / Н. Бишевець, Н. Гончарова, Гончарук // Вісник – 2020. – № 10 (126). – С. 126–133. DOI: 10.5281/zenodo.4506637

13. Бишевець Н. Інноваційні підходи до удосконалення освітнього процесу майбутніх фахівців з фізичної культури і спорту / Н. Бишевець, Н. Гончарова, М. Родіоненко // Теорія і методика фіз. виховання і спорту. – 2020. – 4. – С. 78–85. DOI: 10.32652/tmfvs.2020.4.78–85

14. Бишевець Н. Підготовка майбутніх фахівців із рекреації та туризму нової формації / Н. Бишевець, Н. Гончарова, К. Сергієнко // Мат. XII Міжнар. наук.-практ. конф. «Проблеми активізації рекреаційно-оздоровчої діяльності населення». – Л. – 2020. – С. 304–308.

15. Бишевець Н. Г. Теорія ймовірностей та математична статистика з використанням табличного процесора MS Excel: навч. посіб. / Н. Г. Бишевець, Н. В. Омецинська, Т. В. Юсипів. – Одеса: Вид. дім «Гельветика», 2021. – 234 с.

16. Бишевець Н. Г. Порівняльний аналіз у науково-спортивній діяльності / Н. Г. Бишевець, І. В. Синіговець, Р. В. Олійник // Вісн. Чернігів. держ. пед. ун-ту Т. Г. Шевченка. – Чернігів: ЧДПУ, 2011. – Т.1. – Вип. 86. – С. 23–28.

17. Бишевець Н. Г. Удосконалення викладання дисципліни «Інноваційні та інформаційні технології в фізичній культурі і спорті» / Н. Г. Бишевець, К. М. Сергієнко, Н. М. Гончарова // Мат. III Всеукр. електрон. наук.-практ. конф. з міжнар. участю. – К.: НУФВСУ, 2020. – С. 51, 52.

18. Бишевець Н. Оптимізаційні задачі в структурі освітнього процесу закладів вищої освіти з фізичної культури і спорту / Н. Бишевець, Н. Гончарова, О. Яковенко, М. Родіоненко // Фіз. виховання, спорт і культура здоров'я у суч. сусп-ві. – 2020. – № 2 (50). – С. 3–12. DOI: https://doi.org/10.29038/2220–7481–2020–02–03–12

19. Бишевець Н. Г. Сучасні методи аналізу даних в спорті на прикладі показників довжини тіла чоловічих збірних команд світу з волейболу / Н. Г. Бишевець, К. М. Сергієнко, І. В. Синіговець, В. С. Бровіна // Вісн. Чернігів. держ. пед. ун-ту Т. Г. Шевченка. – Чернігів: ЧДПУ, 2010. – № 81. – С. 151–155.

20. Бишевець Н. Г. Економетричне моделювання та прогнозування в Excel: навч. посіб. / Н. Г. Бишевець, А. І. Кузьмичов та ін.; ред. А. І. Кузьмичов // Акад. муніцип. упр. – К., 2010. – 324 с.

21. Блистів Т. Використання інформаційних технологій при складанні маршруту туристських мандрівок. Проблеми активізації рекреаційно-оздоровчої діяльності населення: Мат. VII Всеукр. наук.-практ конф. з міжнар. участю / Т. Блистів, К. Сергієнко, Н. Бишевець. – Л., 2010. С. 329–335.

22. Герасименко С. Удосконалення навчального процесу у вищих навчальних закладах фізкультурного профілю на основі викладання природничо-наукових дисциплін / С. Герасименко, Н. Бишевець // Теорія і методика фіз. виховання і спорту. – 2006. – № 4. – С. 99–100.

23. Денисова Л. Зарубіжний досвід організаційно-управлінського забезпечення сфери фізичної культури і спорту в умовах цифрової трансформації / Денисова Л., Лавров В., Шинкарук О. // Sport Science Spectrum. – 2024. – № 2. С. 65–73.

24. Денисова Л. В. Алгоритм аналізу анкетних даних у спортивно–педагогічних дослідженнях / Л. В. Денисова, В. В. Усиченко, Н. Г. Бишевець // Педагогіка, психологія та медико-біологічні проблеми фіз. виховання і спорту. – Х.: ХДАДМ (ХХПІ). – 2012. – № 1. – С. 56–60.

25. Додонов О. Г. Оптимізаційні моделі еволюційного програмування в MS Excel: розв'язанні задачі комівояжера з обмеженнями alldifferent / О. Г. Додонов, А. М. Кузьмичов // Реєстрація, зберігання і обробка даних. – 2011. – № 3 (13). – С. 3–16.

26. Іващук О. Т. Економіко-математичне моделювання: навч. посіб. – Т.: ТНЕУ «Економ. думка», 2008. – 704 с.

27. Іксанов О. М. Застосування задач лінійного програмування при розробці інтерактивного атласу Києва / О. М. Іксанов, С. В. Полоцький, О. Г. Голубцов // Фіз. географія та геоморфологія. – 2016. – Вип. 4 (84). – С. 117–120.

28. Кузьмичов А. І. Ймовірнісне та статистичне моделювання в Excel для прийняття рішень: навч. посіб. 4-е вид. / А. І. Кузьмичов, Г. В. Куценко, Н. Г. Бишевець та ін. – К.: Вид-во Ліра. – К., 2020. – 200 с.

29. Левченко А. Ю. Методи прискорювання обчислень в задачах оптимальної маршрутизації: дис. ... канд. техн. наук: 01.05.02. / А. Ю. Левченко. Х.: Харків. нац. ун-т радіоелектроніки, 2013. – 20 с.

30. Овчарук І. Методики розв'язання задач лінійного програмування з використанням сучасних комп'ютерних технологій / І. Овчарук, В. Овчарук // Цифрова платформа: інформаційні технології в соціокультурній сфері. – 2018. – № 2. – С. 73–81.

31. Сергієнко К. Оптимізація етапів прийняття управлінських рішень в системі підготовки висококваліфікованих спортсменів / К. Сергієнко, Н. Бишевець, Л. Богачук, А. Жирнов // Спорт. вісн. Придніпров'я. – 2010. – № 3. – С. 7–10.

32. Тимофієва Н. К. Про деякі властивості множини розв'язків задачі комівояжера / Н. К. Тимофієва // Упр. системы и машины. – 2018. – № 5. – С. 3–12. DOI: 10.15407/usim.2018.05.003

33. Усиченко В. В. Досвід використання баз даних при розробці комп'ютерної програми «Атлет» для спортсменів, які спеціалізуються з бодібілдингу / В. В. Усиченко, Н. Г. Бишевець // Теорія і методика фіз. виховання і спорту. – 2010. – № 3. – С. 67–70.

34. Усиченко В. В. Статистична вірогідність результатів вимірів у спортивно-педагогічній практиці при малій кількості випробувань / В. В. Усиченко, А. М. Лапутін, Н. Г. Бишевець // Педагогіка, психологія та медико-біологічні проблеми фіз. виховання і спорту: зб. наук. пр. за ред. С. С. Єрмакова – Х.: ХДАДМ (ХХПІ). – 2006. – № 11. – С. 105–107.

35. Шинкарук О. Інформаційні технології та кіберспорт: інноваційний підхід до реабілітації військовослужбовців та ветеранів війни: навчально-методичний посібник / Шинкарук О., Бишевець Н., Сергієнко К. та ін. – К., 2024. — 174 с.

36. Шинкарук О. Технологія проектування інформаційного середовища закладу вищої освіти з фізичної культури і спорту : монографія / Шинкарук О., Бишевець Н., Сергієнко К., Яковенко О. – К: Олімп. л-ра, 2021. – 220 с.

37. Шинкарук О. Управління пізнавальною діяльністю студентів у інформаційно-освітньому середовищі закладу вищої освіти спортивного профілю / Шинкарук О., Бишевець Н., Сергієнко К. та ін. // Фізична культура, спорт та здоров'я нації: зб. наук. праць. – Вип. 11. – С 77–87. DOI: 10.31652/2071-5285-2021-11(30)-77-87

38. Byshevets N. Training of higher educational institution's students for perfor-ming engineering designs / N. Byshevets // Інформ. техн. в освіті. – 2016. – N 2 (27). – С. 154–166.

39. Byshevets N. Development skills implementation of analysis of variance at sport-pedagogical and biomedical researches / N. Byshevets, O. Shynkaruk, O. Stepanenko et al // J. of Physical Education and Sport. – 2019. – Vol. 19 (311). – P. 2086-2090. https://doi.org/10.7752/jpes.2019.s6311.

40. Byshevets N. Using the methods of mathematical statistics in sports and educational research of masters in physical education and sport / N. Byshevets, L. Denysova, O. Shynkaruk et al // J. of Physical Education and Sport. – 2019. –19 (148). – P. 1030-1034. DOI:10.7752/jpes.2019.s3148

41. Byshevets N. Formation of the Knowledge and Skills to Apply Non–Parametric Methods of Data Analysis in Future Specialists of Physical Education and Sports / N. Byshevets,

O. Iakovenko, O. Stepanenko et al // Sport Mont. – 2021. – Vol. 19, N S2. – P. 171-175. DOI: 10.26773/smj.210929.

42. Gayev Y. The travelling salesman problem in the engineering education programming curriculum / Y. Gayev, V. Kalmikov // Proceedings of the National Aviation University. – 2017. – N 3(72). – P. 90-98.

43. Kashuba V. The Formation of Human Movement and Sports Skills in Processing Sports–lpedagogical and Biomedical Data in Masters of Sports / V. Kashuba, O. Stepanenko, N. Byshevets et al // International J. of Human Movement and Sports Scie. – 2020. – N 8 (5). – P. 249-257. DOI: 10.13189/saj.2020.080513.

44. McCarthy J. Automatic programming. Information Processing / J. McCarthy. –1956. – N 1. – P. 10-12.

45. Shynkaruk O. et al. Modern approaches to the preparation system of Masters in eSports / O. Shynkaruk, N. Byshevets, O. Iakovenko, et al // Sport Mont. – 2021. 19.– S2. – P. 69-74. https://doi.org/10.26773/smj.210912.

Навчальне видання

ШИНКАРУК Оксана
БИШЕВЕЦЬ Наталія
СЕРГІЄНКО Костянтин
ЯКОВЕНКО Олена
УСИЧЕНКО Віталій

# ОСНОВИ ПРОГРАМУВАННЯ, СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОБУДОВА КОМП'ЮТЕРНИХ СИСТЕМ

Редагування – НАДІЯ ОТРОХ
Комп'ютерна графіка – ОКСАНА ЦІКАЛО
Комп'ютерне верстання – АЛЛА КОРКІШКО
Технічне редагування – ТЕТЯНА БЕРЕЗЯК

Формат 60×90/16. Ум.-друк. арк. 8,75.
Тираж 50 пр. Зам. №

Національний університет
фізичного виховання і спорту України,
видавництво «Олімпійська література»
Україна, 03150, Київ-150, вул. Фізкультури, 1