Міністерство освіти України

Державний університет

“Львівська політехніка”

Кафедра ІСМ

**КУРСОВА РОБОТА**

з предмету “Методи підтримки прийняття рішень”

на тему

***“Модель колективного вибору  
рішень”***

Виконала:

студентка гр. ІСМ-5М

Шаховська Наталя

Залікова книга: № 9517007

Перевірив:

доц. Катренко А. В.

Львів – 1999

Кафедра “Інформаційні системи та мережі”

Фах “Інтелектуальні системи прийняття рішень”

Базовий напрямок “Комп’ютерні науки”

ЗАВДАННЯ НА КУРСОВУ РОБОТУ

з предмету “Методи підтримки прийняття рішень”

студентки гр. ІСМ-5М

Шаховської Наталі

1. Тема: “Модель колективного вибору рішень”
2. Завдання: розробити програму для демонстрації роботи одного з методів голосування.
3. Зміст пояснювальної записки.

1 Змістовна постановка задачі

2 Формальна постановка задачі

3 Математичні методи розв’язку

4 Опис алгоритму

4.1 Визначення переможця Борда

4.2 знаходження оцінки Копленда

4.3 Алгоритм визначення переможця за правилами Борда чи Копленда

5 Опис програми

5.1 Вибір технології програмування

5.2 Структура програми

5.3 Інструкція користувачеві

6 Контрольний приклад

Висновки

1. Перелік графічного матеріалу.

Кількість малюнків – 5.

Завдання видане: 10.09.99

Завдання видав: доц. Катренко А. В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Завдання прийняла: Шаховська Наталя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Львів – 99**ЗМІСТ**

Вступ 4

1 Змістовна постановка задачі 6

2 Формальна постановка задачі 10

3 Математичні методи розв’язку 18

4 Опис алгоритму 23

4.1 Визначення переможця Борда 23

4.2 Знаходження оцінки Копленда 25

4.3 Алгоритм визначення переможця за правилами Борда чи Копленда 28

5 Опис програми 31

5.1 Вибір технології програмування 31

5.2 Структура програми 33

5.3 Інструкція користувачеві 35

6 Контрольний приклад 37

Висновки 39

Список літератури 40

Додатки 41

Програма 41

Результати роботи програми 45

ВСТУП

|  |
| --- |
| “Демократія як метод керування використовує результати суспільних рішень громадян на виборах і рішень законодавців у представницьких орга­нах”  *(Рікер [1982]).* |

Більшість суспільних розподілених рішень (таких, як податки і суспільні витрати) приймається на основі голосування. Вибори також використовуються для поповнення багатьох суспільних закладів. Тут ми маємо важливі приклади чистих суспільних продуктів (наприклад, усі громадяни даного міста без яких-небудь винятків беруть участь у “споживанні” свого мера), що вибираються на основі голосування і без побічних платежів.

Починаючи з політичної філософії Просвітництва, вибір правил голосування був головною етичною проблемою, пов'язаною з додатками, що далеко йдуть, для функціонування більшості політичних інститутів. Дебати про справедливість різноманітних методів голосування почалися з досліджень де Борда [1781] і Кондорсе [1785]. У 1952 році Ерроу запропонував формальну модель, що протягом трьох десятиліть аналізувалася в численних роботах математичної орієнтації по так званому колективному виборі.

Формально правило голосування вирішує задачу колективного ухвалення рішення, у котрої декілька індивідуальних агентів (виборців) повинні спільно вибрати один із декількох результатів (також званих кандидатами), щодо котрих їхні думки розходяться. Будемо припускати, що кінцева множина *N* виборців повинна обрати одного кандидата з кінцевої множини *А*. Для простоти припустимо, що індивідуальні думки (або переваги) не припускають випадків байдужності. Кожна така перевага є довільним лінійним порядком на *А*.

Правило голосування вибирає кандидата на основі повідомлених порядкових переваг і тільки на основі цих переваг. У цьому істотна відмінність від моделей, у яких гроші й інші продукти дозволяли здійснювати довільно малі компенсації для агентів. Голосування не припускає поступки між двома кандидатами інакше, ніж за рахунок можливого обрання третього кандидата.

Якщо кандидатів тільки два, то звичайне правило голосування більшістю голосів безперечно є найбільш справедливим методом. Цей принцип більшості - вихідний пункт процесу демократичного прийняття рішень. Він був явно сформульований два сторіччя тому, а його основа є набагато більш древньою. Аксіоматична формалізація принципу більшості запропонована Меєм.

Розгляду методів голосування і втіленню у програму одного з них і присвячена дана курсова робота. Буде проведена порівняльна характеристика різних методів голосування, і за допомогою контрольного прикладу продемонстрована робота одного з них.

# ЗМІСТОВНА ПОСТАНОВКА ЗАДАЧІ

Завдання, яке ставиться переді мною у даній курсовій роботі – забезпечити процес виборів, тобто кінцева множина *N* виборців повинна обрати одного кандидата з кінцевої множини *А*. Обов’язковою умовою є обрання *єдиного* кандидата. Для простоти припустимо, що індивідуальні думки (або переваги) не припускають випадків байдужності. Кожна така перевага є довільним лінійним порядом на *А* (тобто повне транзитивне й асиметричне бінарне відношення). Це припущення не призводить до істотних втрат загальності.

Формально правило голосування вирішує задачу колективного ухвалення рішення, у котрої декілька індивідуальних агентів (виборців) повинні спільно вибрати один із декількох результатів (також званих кандидатами), щодо котрих їхні думки розходяться.

Правило голосування являє собою систематичне рішення, яке у всій повноті опирається на індивідуальні думки. Позначимо через *L*(*А*) множину лінійних порядків на *А*,тоді правило голосування є відображенням *L*(*А*)*N* в *А*. Те, що правило голосування може бути визначене для будь-якої мислимої конфігурації переваг, виражає фундаментальний принцип свободи думок: кожний виборець має право ранджувати кандидатів будь-яким чином. Проте в деяких моделях голосування, що містять економічні змінні або невизначені результати, можна припускати, що переваги виборців задовольняють деякій загальній умові. Це особливо зручно при стратегічному аналізі голосування і при агрегуванню переваг.

Правило голосування вибирає кандидата на основі повідомлених порядкових переваг і тільки на основі цих переваг. У цьому істотна відмінність від моделей, у яких гроші й інші продукти дозволяли здійснювати довільно малі компенсації для агентів. Голосування не припускає поступки між двома кандидатами інакше, ніж за рахунок можливого обрання третього кандидата.

Нехай дано як контрольний приклад наступний профіль для 9 виборців і 5-ти кандидатів:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 4 | 1 | 3 |
| a  b  c  d  e | c  d  b  e  a | e  a  d  b  c | e  a  b  d  c |

У кожному стовпці кандидати розташовані у порядку зменшення їх значущості для кожної групи виборців. Тобто, для першого стовпця (групи виборців, що складається з однієї людини) можна визначити переваги наступним чином: група виборців, що складається з однієї особи, вважає кандидата *a* найкращим. На другому місці вони ставлять кандидата *b*, на третьому місці *c* і т.д. аналогічно кандидати ранджовані у кожній групі.

Завдання: визначити єдиного переможця виборів.

Існує багато способів визначення переможця. Вони будуть описані і відповідним чином порівняні у наступних розділах.

Зазначимо зараз, що дана курсова робота присвячена розгляду і втіленню у програму метода Копленда і порівнянню отриманого результату із результатом за методом Борда.

Визначимо **правило Копленда**.

Порівняємо кандидата *a* з будь-яким іншим кандидатом *x*. Нарахуємо йому +1, якщо для більшості *a* краще за *x*, -1, якщо для більшості *x* краще за *a*, і 0 при всіх *x*, *x*≠*a*, одержуємо оцінку Копленда для *a*. Обираємо кандидат, названий переможцем за Коплендом, із найвищою такою оцінкою [див. 1, ст. 321].

В даному правилі не вказано, що робити у тому випадку, коли знайдуться два або більше кандидати з однаковою оцінкою Копленда. Припустимо, що обереться той кандидат, ім’я і прізвище якого стоїть найближче за списком. Це припущення порушує правило нейтральності [див. 1, ст. 329], але, як буде доведено у наступних розділах, кожне правило голосування Копленда є найбільш наочним і легким для комп’ютерної реалізації.

**Правило Борда:** кожний виборець повідомляє свої переваги, ранджуючи *p* кандидатів від найкращого до найгіршого (байдужість забороняється). Кандидат не одержує очок за останнє місце, одержує одне очко за передостаннє місце і т.д., одержує *p*-1 очко за перше місце. Перемагає кандидат із найбільшою сумою очок. Він називається переможцем за Борда.

Тут так само не вказується, що робити при рівності очок, тобто також може порушуватись умова нейтральності.

Охарактеризуємо вище поставлену задачу.

Її критерієм якості є максимізація оцінки Копледа (Борда).

Обмеженнями виступають переваги виборців і їх ранджування кандидатів. Як буде вказано далі, фактично потрібно накладати також обмеження і на кількість виборців та кількість кандидатів. Проте це обмеження не є суттєвим, так як завжди при голосуванні можна провести поділ на округи.

За рівнем складності це є задача Р-типу. Час розв’язку даної задачі становить t0=a0+a1x+… +anxn і залежить від кількості груп виборців та кількості кандидатів.

Знаходження переможця за правилами Копленда і Борда є найпростішими за своєю структурою, оптимальні за Парето, анонімні, нейтральні (якщо не вказувати, що робити при рівності очок). Крім того, правило Борда иакож задовольняє аксіомі участі і поповнення (вони будуть розглянуті у наступному розділі).

**Оптимальність за Парето:**

Якщо кандидат *a* для всіх кращий від кандидата *b*, то *b* не може бути обраний.

**Анонімність:**

Імена виборців не мають значення – якщо два виборці поміняються голосами, то результат виборів не зміниться.

**Нейтральність:**

Імена кандидатів не мають значення. Якщо ми поміняємо місцями кандидатів а і b у перевазі кожного виборця, то результат голосування зміниться відповідно (якщо раніш вибирався *а*, то тепер буде вибиратися *b* і навпаки; якщо вибирався деякий *х*, відмінний від *а* і *b*, то він же і буде обраний).

**Монотонність:**

Припустимо, що *а* вибирається (серед переможців) при даному профілі і профіль змінюється тільки так, що положення *а* поліпшується, а відносне порівняння пари будь-яких інших кандидатів для будь-якого виборця залишається незмінним. Тоді *а* як і раніше буде обраний (знову серед переможців) для нового профілю.

# Формальна постановка задачі

Приведемо ще раз задачу даної курсової роботи: використовуючи профіль переваг виборців, визначити єдиного переможця з множини заданих. Повинна існувати можливість перевірки коректності задання профілю. Обмеженнями на задачу є відсутність байдужості та ранжування кандидатів у строгому порядку.

Опишемо методи голосування, які можуть використовуватись для розв’язання даної задачі і наведемо ряд основних визначень і теорем.

*Правило відносної більшості*. Кожний виборець віддає свої голос найкращому для себе кандидату. Обирається кандидат, згаданий у найбільшій кількості бюлетенів. Це правило може суперечити думці більшості (див. 1, приклад 9.1).

**Визначення 2.1. Правило Борда.** Кожний виборець повідомляє свої переваги, ранжуючи *р* кандидатів від кращого до гіршого (байдужність забороняється). Кандидат не одержує очок за останнє місце, одержує одне очко за передостаннє і так далі, одержує *р*-1 очок за перше місце. Перемагає кандидат із найбільшою **сумою** очок. Він називається **переможцем за Борда.**

Ми не уточнюємо, що робити при рівності очок.

**Визначення 2.2.** *Для заданого профілю переваг* ***переможцем за Кондорсе*** *називається кандидат а* (*із необхідністю єдиний*), *що перемагає будь-якого іншого кандидата при парному порівнянні за правилом більшості:*

*для всякого b≠а виборців*, *що вважають а кращим за b*, *більше*, *ніж тих*, *хто вважає*, *що b кращим за а*.

*Заможне за Кондорсе правило вибирає переможця за Кондорсе*, *якщо такий існує*.

Відсутність переможця за Кондорсе є знаменитим “парадоксом голосування”. Як часто може спостерігатися парадокс голосування? У загальному випадку ймовірність *π*(*p*, *n*) того, що переможця за Кондорсе не існує при *р* кандидатах і *п* виборцях, зростає по *р* і зростає по числу виборців від *п* до *n*+2. Це може бути перевірене на основі обчислення *π*(*п*, *р)* для малих значень *п* і *р*, але в загальному випадку це твердження залишається недоведеним припущенням.

Парадокс голосування стає майже достовірною подією, коли число кандидатів стає достатньо великим при фіксованому *п*. Якщо число виборців стає достатньо великим при фіксованому *р*, то гранична ймовірність *π*(*p*) може бути оцінена за Фішберн [1984]:



яка справедлива з точністю до половини відсотка при *р*≤50.

**Визначення 2.3. Правила голосування з підрахунком очок.**

*Фіксуємо послідовність дійсних чисел , що не спадає*

s0≤s1≤…≤sp-1 при s0<sp-1.

*Виборці ранжують кандидатів*, *причому s0 очок дається за останнє місце*, *s1 - за передостаннє і так далі*. *Обирається кандидат із максимальною сумою очок*.

**Визначення 2.4. Правило Копленда.** *Порівняємо кандидата а з будь-яким іншим кандидатом х*. *Нарахуємо йому* +1, *якщо для більшості а краще за х*,-1, *якщо для більшості х краще за а*, *і 0 при рівності*. *Сумуючи загальну кількість очок по всім х*, *х*≠*а*, *одержуємо оцінку Копленда для а*. *Обирається кандидат*, *названий***переможцем за Коплендом,** *із найвищою з таких оцінок*.

**Визначення 2.5. Правило Сімпсона.** *Розглянемо кандидата а*, *будь-якого іншого кандидата х і позначимо через N*(*a*,*x*) *число виборців*, *для котрих а краще за х*. *Оцінкою Сімпсона для а називається мінімальне з чисел N*(*a*,*x*) *по всім х*, *х*≠*а*. *Обирається кандидат*, *названий***переможцем за Сімпсоном,** *із найвищою такою оцінкою*.

Обидва ці правила заможні за Кондорсе.

*Оптимальність за Парето*. Якщо кандидат *а* для всіх кращий від кандидата *b*, то *b* не може бути обраним.

*Анонімність*. Імена виборців не мають значення: якщо два вибореці поміняються голосами, то результат виборів не зміниться.

*Нейтральність*. Імена кандидатів не мають значення. Якщо ми поміняємо місцями кандидатів а і b у перевазі кожного виборця, то результат голосування зміниться відповідно (якщо раніш вибирався *а*, то тепер буде вибиратися *b* і навпаки; якщо вибирався деякий *х*, відмінний від *а* і *b*, то він же і буде обраний).

Правила Копленда і Сімпсона оптимальні за Парето, анонімні і нейтральні, якщо ми розглядаємо їх як відображення, що ставлять у відповідність кожному профілю переваг підмножину переможців. Анонімність і нейтральність очевидні. Перевірити, що множини переможців за Борда (Коплендом, Сімпсоном) містять тільки оптимальні за Парето результати, достатньо просто. Так, оцінка Сімпсона кандидату, що домінується за Парето, дорівнює нулю, а для оптимального за Парето кандидата вона позитивна.

*Монотонність*. Припустимо, що *а* вибирається (серед переможців) при даному профілі і профіль змінюється тільки так, що положення *а* поліпшується, а відносне порівняння пари будь-яких інших кандидатів для будь-якого виборця залишається незмінним. Тоді *а* як і раніше буде обраний (знову серед переможців) для нового профілю.

Всі правила підрахунку очок, а також правила Копленда і Сімпсона є монотонними.

*Відносна більшість із вибуванням*. У першому раунді кожний виборець подає один голос за одного кандидата. Якщо кандидат набирає сувору більшість голосів, то він і обирається. У противному випадку в другому турі проводиться голосування за правилом більшості з двома кандидатами, що набрали найбільшу кількість голосів у першому турі.

Прихильники цього методу підтверджують, що він майже так само простий, як і правило відносної більшості (виборцям не потрібно повідомляти повне ранжування кандидатів), і виключає марнотратні вибори. При звичайному правилі відносної більшості, якщо я голосую за кандидата, що одержує маленьку підтримку, то мій голос буде марним. Проте при вибуванні в мене є ще один шанс вплинути на результат. Проте цей метод *не є* монотонним, як показують такі два профілі з 17 виборцями:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Профіль *А* | | | |  | Профіль *B* | | | |
| 6 | 5 | 4 | 2 |  | 6 | 5 | 4 | 2 |
| *a* | *c* | *b* | *b* |  | *a* | *c* | *b* | *a* |
| *b* | *a* | *c* | *a* |  | *b* | *a* | *c* | *b* |
| *c* | *b* | *a* | *c* |  | *c* | *b* | *a* | *c* |

При профілі *А* в другий тур проходять *а* і *b* і виграє *а* (11 голосів проти 6). Профіль *В* такий же за одним винятком. У двох виборців перевага *b*>*a*>*с* змінюється на перевагу *а*>*b*>*с*, тобто для них тепер *а* краще за *b*.Тепер у другий тур проходять *а* і *с*, причому виграє *с* (9 голосів проти 8). Таким чином, поліпшення позиції кандидата *а* призводить до його поразки!

*Метод альтернативних голосів*. Виключимо спочатку тих, хто одержав найменшу кількість голосів. Потім порахуємо голоси для кандидатів, що залишилися, і знову виключимо невдах. Будемо повторювати цю операцію доти, поки не залишиться один кандидат (або множина кандидатів із рівним числом голосів).

Тут головна увага приділяється тому, щоб не загубити ніякі голоси і кожному дати шанс підтримати кандидата, який подобається найбільше. У цьому підході повторно використовуються методи підрахунку очок для винятку кандидатів-невдах. На жаль, *будь-яке* правило, засноване на послідовному виключенні за методом підрахунку очок, повинне порушувати властивість монотонності для деяких профілів.

*Поповнення* (*однозначні правила голосування*). Дві групи виборців N1*,* N2, що не *перетинаються,* мають справу з тією ж множиною *А* кандидатів. Нехай виборці *N1* і *N2* вибирають того самого кандидата *а*.Тоді виборці *N1*∪*N2* також оберуть *а* з *А.*

Ця властивість є дуже обгрунтованою, коли єдиний виборчий орган розбитий на велику кількість підмножин, як у випадку регіональних асамблей і підкомітетів.

*Поповнення* (*відображення голосування*). Дві групи виборців N1*,* N2, що не *перетинаються,* мають справу з тією ж множиною *А* кандидатів. Нехай виборці *Ni* обирають підмножину *Вi* з *А* при *i*=1,2. Якщо *В*1 і *B2* перетинаються, то виборці *N1*∪*N2* оберуть *В1*∩*B2* як множину найкращих для себе результатів.

**Теорема 2.1** (Янг [1975])

(*a*) *Всі відображення голосування*, *засновані на підрахунку очок* (*підмножини кандидатів, що вибирають, із найбільшою сумарною кількістю* очок)*, задовольняють аксіомі* поповнення. *Якщо при рівності очок вибір проводиться на основі фіксованого порядку на А*, *то відповідні правила голосування також задовольняють аксіомі поповнення*.

(*b*) *Не існує заможного за Кондорсе правила голосування* (*або відображення голосування*), *яке б задовольняло аксіомі поповнення*.

*Аксіома участі*. Нехай кандидат *а* вибирається з множини *А* виборцями з *N*. Розглянемо далі виборця *i* поза *N*. Тоді виборці з *N*∪{*i*} повинні обрати або *а*, або кандидата, що для агента *i* строго краще *а*.

Це означає, що якщо додатковий голос дійсно змінює результат виборів, то це може бути тільки на руку “ключовому” виборцю.

**Теорема 2.2** (Мулен [1986с])

(*a*) *Для всіх правил голосування з підрахунком очок*, *коли при рівності очок вибір здійснюється за допомогою заданого порядку на А*, *виконується аксіома участі*.

(*b*) *Якщо А складається хоча б із чотирьох кандидатів*, *то жодне заможне за Кондорсе правило голосування не задовольняє аксіомі участі*.

*Безперервність*. Нехай виборці з *N1* обирають кандидата *а* з *A*, а група *N2*, що не пересікається з *N1*, обирає іншого кандидата *b*. Тоді існує достатньо велике число *m* дублів групи виборців *N1*, таке що комбінована група виборців (*mN1*)∪*N2*обере *а*.

**Теорема 2.3** (*Янг [1975]*). *Відображення голосування засноване на методі підрахунку очок* (*визначення 2.3 без фіксації правила для випадку рівності очок*) *тоді і тільки тоді*, *коли воно задовольняє таким чотирьом властивостям:*

*анонімність*, *нейтральність*,

*аксіома поповнення і безперервність*.

b

a

c

d

*Голосування з послідовним винятком*. Спочатку за правилом більшості виключається або *а*, або *b*,потім за правилом більшості проводиться порівняння переможця першого раунду і *с* і так далі. У випадку рівності програє нижній кандидат.

У цьому процесі поправок нехай *а -* поправка, *b -* поправка до поправки, *с -* вихідна пропозиція, *d - status quo*.

Цей метод задовольняє аксіомі спроможністі за Кондорсе: якщо *а -* переможець за Кондорсе, то він виграє. Насправді спроможність при порівняннях за правилом більшості справедлива в ширшому змісті.

*Спроможність за Смітом*. Якщо множина *А* кандидатів розбивається на дві підмножини *В*1, *B2*, що не перетинаються, і кожний кандидат *b1*∈*В*1 виграє (за суворою більшістю) у будь-якого кандидата *b2*∈*В*2, то повинний бути обраний результат із *В*1.

З іншого боку, голосування при послідовному винятку очевидно не є нейтральним. Порядок виключень, звичайно, впливає на результат.

*Правило рівнобіжного виключення*. Спочатку за правилом більшості дорівнюються пари *а* з *b* і *с* з *d*. Переможці зустрічаються у фіналі, де порівнюються за правилом більшості. У випадку рівності вибирається кандидат, що йде раніше за алфавітом.

Це - знову заможний за Кондорсе метод. Більш того, для обрання кожному кандидату *х* потрібно перемогти в двох порівняннях за правилом більшості. Припустимо спочатку, що рівності при порівнянні з цими двома кандидатами немає (*х* виграє для суворої більшості). Тоді *х* не може домінуватися за Парето деяким кандидатом *у*, інакше *b* був би переможцем за Кондорсе. Отже, метод рівнобіжного виключення вибирає оптимальний за Парето результат у (найбільше поширеному) випадку, коли при бінарних виборах немає рівностей. Проте якщо рівності можливі, то оптимальність за Парето може порушуватися.

*Бінарним деревом* на *А* є таке кінцеве дерево, у котрому кожній нефінальній вершині (включаючи початкову) відповідають рівно дві наступні, а кожній фінальній вершині (у котрої немає наступних) приписаний кандидат (елемент із *A*), причому кожний кандидат з'являється принаймні в одній фінальній вершині.

Серед бінарних дерев найпростішими є ті, у котрих кожний кандидат приписаний *рівно одній* вершині. Назвемо їх *деревами без повторних виключень*.

**Лема 2.1**

(*а)* *Якщо А складається з трьох кандидатів*, *то дерево після послідовного виключення є єдиним безповторним деревом*. *Відповідне правило голосування оптимальне за Парето* (*при нашій умові*, *що всі порівняння по більшості суворі*).

(*b*) *Якщо А складається з чотирьох кандидатів*, *тобто тільки два безповторних дерева: послідовне виключення і рівнобіжне виключення*. *Перше з них порушує оптимальність за Парето*, *а останнє - ні*.

(*c*) *Якщо А містить п'ять або більше кандидатів*, *то будь-яке виключення по безповторному дереву призводить до обрання кандидата для деяких профілі,в що домінується за Парето*.

Існує бінарне дерево, визначене для довільної кількості учасників, що дозволяє уникнути обох цих небезпек. Відповідні послідовні *виключення* породжують оптимальне за Парето, анонімне і монотонне правило голосування. Це дерево називається деревом *багатоетапного виключення*.

Для кожного конкретного упорядкування кандидатів існує по одному такому дереву. Позначимо через *Гp*(1,2,... ,*р)* дерево, що відповідає порядку *A*={1,2,... ,*р*}. Визначимо його індуктивно за розміром *А:*

1

2

Г*p*(1,2,...,(*р*-2),*p*)

Г*p*(1,2,...,*р*-1)

Г*p*(1,2,...,*р*)

Г2(1,2)

Так, для трьох і чотирьох кандидатів одержуємо:

1

3

2

1

1

1

2

1

3

2

4

1

Г4(1,2,3)

Г4(1,2,3,4)

При *р* кандидатах утворюються 2p-l фінальні вершини; кандидат 1 приписаний 2p-2 фінальним вершинам, а кандидат *р* тільки однієї. Тим не менше для обрання навіть кандидату *р* потрібно перемогти в *р*-1 дуелях (хоча йому можливо прийдеться по декілька разів зіткнутися з тим самим опонентом).

Хоча дерево багатоетапного виключення велике, його рішення (тобто обчислення кандидата, що виграє) може бути отримане за допомогою дуже простого алгоритму

**Теорема 2.4** (Шепсл і Вейнгаст [1984]).

Задані дерево багатоетапного виключення Гp(1,2,... ,*р)* і профіль переваги, що відповідає мажоритарному турніру Т. Кандидат *а*\* може бути знайдений за таким алгоритмом:

(12)



**Наслідок теореми 2.4.** *Кандидат а*, що *вибирається по дереву багатоетапного виключення з турніром Т*, *задовольняє умові:*

*для будь-якого b*∈*А*, *b*≠*а:*

{*аТb*} *і/або* {*для деякого с*, *аТс і сTb*}. (14)

*Зокрема, а оптимальний за Парето*. *Більш того*, *дерево багатоетапного виключення породжує монотонний метод голосування*.

Серед заможних за Кондорсе правил голосування ми виявили три методи, що задовольняють основним вимогам оптимальності за Парето, анонімності і монотонності: множина переможців за Коплендом, множина переможців за Сімпсоном і дерево багатоетапного виключення. Перші два нейтральні, але можуть виділяти декількох переможців (додаткове правило при рівності очок порушить нейтральність). Зауважимо, що переможець при багатоетапному виключенні знаходиться швидше, оскільки алгоритм (12) у середньому потребує порівняння не більше половини від усіх *p*(*p-l*) пар. У той же час для визначення переможців за Коплендом і Сімпсоном потрібно провести весь турнір порівнянь за правилом більшості.



# Математичні методи розв’язку

У попередньому розділі були описані методи підрахунку очок і основні вимоги до них. Порівняємо ці методи.

Як було зазначено, найлегшим серед них, але й найгіршим, є правило відносної більшості. Можна переконатись, що насправді воно суперечить думці більшості. Тому воно не може бути вибране для комп’ютерної реалізації.

Як Борда, так і Кондорсе зауважили, що правило відносної більшості може призводити до обрання поганого кандидата, точніше такого кандидата, що у парному порівнянні за правилом більшості програє *будь-якому* іншому кандидату.

Для того щоб перебороти цю хибу, Кондорсе і Борда запропонували відмовитися від правила відносної більшості, причому кожний із них запропонував своє правило замість даного. Кондорсе запропонував вибирати кандидата, що перемагає будь-якого іншого кандидата в парному порівнянні, якщо такий *переможець за Кондорсе* існує. Борда запропонував приписати очко кожному кандидату, який лінійно зростає у залежності від його рангу в перевазі виборця. Потім він запропонував обрати кандидата, що одержав найбільшу сумарну кількість очок в усіх виборців. Ці дві ідеї породжують два найбільше важливих сімейств правил голосування.

Результати цих правил голосування можуть сильно відрізнятися за властивостями. Однією із таких властивостей є аксіома монотонності. Правило голосування називається монотонним, якщо кандидат *залишається* обраним при посиленні його підтримки (тобто коли відносна позиція даного кандидата у чиїхось перевагах поліпшується, а відносні позиції інших кандидатів не змінюються). Всі методи підрахунку очок є монотонними, але деякі відомі методи, що виникають із підрахунку очок, не є такими. Прикладами таких правил служать дуже популярне правило відносної більшості з вибуванням і метод альтернативних голосів.

Є дві аксіоми, що призводять до критики заможних за Кондорсе правил (оскільки дані правила порушують ці аксіоми). З іншого боку, на цих аксіомах заснована характеризація методу підрахунку очок. Ці аксіоми порівнюють кандидатів, обраних різноманітними групами виборців. Вони називаються властивостями поповнення й участі. Поповнення означає, що якщо дві групи виборців , що не перетинаються, (наприклад, сенат і палата представників) обирають того самого кандидата *а*, то об'єднання цих двох виборчих органів підтвердить обрання *а*. Участь означає, що виборець не може виграти, утримуючись від голосування, у порівнянні з можливістю брати участь у голосуванні і висловити свої переваги. *Будь-яке* заможне за Кондорсе правило порушує обидві ці аксіоми. На противагу цьому правила підрахунку очок характеризуються властивістю поповнення (теорема Янга) і задовольняють аксіомі участі. Теорема Янга в даний час є найістотнішим доказом у підтримку методів підрахунку очок, зокрема системи очок Борда.

Заможні за Кондорсе правила голосування усе ж надзвичайно популярні, зокрема, завдяки простоті доведення парного порівняння за правилом більшості. Відповідний клас заможних за Кондорсе методів заснований на послідовних порівняннях за правилом більшості. Законопроект і численні поправки до нього в конгресі США голосуються саме у такий спосіб. Відомий метод послідовного винятку може порушувати умову оптимальності за Парето. Інші методи, засновані на бінарних деревах парних порівнянь за правилом більшості, суперечать аксіомі монотонності. Найпростіше правило, що засноване на послідовному порівнянні і є оптимальним за Парето і монотонним, називається багатоетапним методом винятку. При використанні цього методу потрібно менше парних порівнянь, ніж в інших, концептуально більш простих методах, наприклад у правилі Копленда. За останнім правилом обирається той, хто виграє більшість парних дуелей. Таким чином, голосування, засноване на послідовних парних порівняннях, може задовольняти найбільше важливим аксіоматичним вимогам, але тільки в тому випадку, якщо ми акуратно виберемо цю послідовність.

Правила Борда, відносної більшості та антибільшості являють собою приклади правил голосування з підрахунком очок. Проте правило антибільшості явно не є монотонне, а правило відносної більшості – несправедливе.

Переможець Борда не може бути найгіршим за Кондорсе, так як він є кандидатом, що має найвищий середній бал. За цим правилом завжди знаходяться оптимальний за Парето переможець або їх множина. Прикладами заможних за Кондорсе правил є правила Копленда та Сімпсона. Так само, як і правило Борда або будь-який інший метод підрахунку очок, ці правила вибирають для кожного профілю підмножину переможців, яка може складатись з декількох кандидатів, що одержали однакову оцінку.

Як вже було зазначено, правила голосування повинні бути монотонні, оптимальні за Парето, антонімні і нейтральні. Всі правила голосування з підрахунком очок, крім правила антибільшості, є оптимальні за Парето, монотонні, анонімні і нейтральні, якщо ми не вказуємо, що робити при рівності очок. Крім того, правила голосування повинні задовольняти аксіомі участі та поповнення. Метод Борда відноситься до цих правил (це було показано у попередньому розділі).

Правила Борда і Копленда, як зазначає Мулен, спираючись на практику, не дуже части призводять до рівності очок [1, ст. 299], тому у цьому ракурсі є найкращими. Проте методи Кондорсе, до яких відноситься і правило Копленда, для деяких профілів може не задовольняти аксіомі участі.

Наступною групою правил є правила, засновані на послідовному виключенні за методом підрахунку очок (відносна більшість з вибуванням, метод альтернативних голосів). Проте ці правила, як і будь-які інші методи з вибуванням кандидатів, порушують властивість монотонності для деяких профілів.

Метод рівнобіжного виключення вибирає оптимальний за Парето результат у (найбільше поширеному) випадку, коли при бінарних виборах немає рівностей. Проте якщо рівності можливі, то оптимальність за Парето може порушуватися.

Незважаючи на вище перераховані труднощі, спроможність за Кондорсе, широко відома у якості демократичного принципу, в той час як правило Борда “приховує” справжні симпатії виборців за математичною формулою.

До заможних до Кондорсе правил відносять також наступні методи голосування:

а) голосування з послідовним винятком. За очевидних причин це правило не є нейтральним і оптимальним за Парето, так як порядок виключень впливає на результат голосування. Визначаючи порядок денний, голова фактично контролює процес виборів. Проте це правило досить широко використовується Конгресом США;

б) правило рівнобіжного виключення. Воно породжує дерево без повторних виключень і вимагає проведення цілої низки мажоритарних турнірів. Як було доведено в попередньому розділі, бінарне дерево може дати оптимальне за Парето правило голосування тільки у складнішому випадку, ніж безповторне дерево. Також може порушуватись монотонність;

в) дерево багатоетапних виключень. Цей метод забезпечує проведення наполовину меншої кількості мажоритарних турнірів, ніж метод Копленда. Воно має великий розмір. Кандидатам, можливо, потрібно брати участь у дуелях з тим самим опонентом по декілька разів. Проте його алгоритм є досить простим. Дерево багатоетапного виключення породжує оптимальний за Парето і монотонний метод голосування. Завжди знаходиться єдиний переможець, а не множина. Проте цей метод породжує всі труднощі, які пов’язані з використанням бінарних дерев.

Таким чином, було проведено порівняльну характеристику усіх методів голосування більшістю голосів із виключенням випадків байдужості і подання неправдивої інформації.

Серед заможних за Кондорсе правил голосування ми виявили три методи, що задовольняють основним вимогам оптимальності за Парето, анонімності і монотонності: множина переможців за Коплендом, множина переможців за Сімпсоном і дерево багатоетапного виключення. Серед методів підрахунку очок найкращим виявився метод визначення переможця за Борда. Заможні за Кондорсе правила застосовані на парному порівнянні кандидатів за правилом відносної більшості. Для пересічного виборця вони є найбільш зрозумілими.

Правило Борда задовольняє аксіомі участі та поповнення, але приховує за математичною формулою справжні переваги виборців.

Для програмної реалізації виберемо один з методів Копленда як найпростіший і для порівняння визначимо переможця за Борда.

Приведемо ще раз правила Копленда і Борда для того, щоб перейти до формулювання алгоритму програми.

**Правило Борда.** Кожний виборець повідомляє свої переваги, ранджуючи *р* кандидатів від кращого до гіршого (байдужність забороняється). Кандидат не одержує очок за останнє місце, одержує одне очко за передостаннє і так далі, одержує *р*-1 очок за перше місце. Перемагає кандидат із найбільшою **сумою** очок. Він називається **переможцем за Борда.**

**Правило Копленда.** Порівняємо кандидата а з будь-яким іншим кандидатом х. Нарахуємо йому +1, якщо для більшості *а* краще за *х*, -1, якщо для більшості *х* краще за *а*, і 0 при рівності. Сумуючи загальну кількість очок по всім *х*, *х≠а*, одержуємо оцінку Копленда для *а*. Обирається кандидат, названий **переможцем за Коплендом,** із найвищою з таких оцінок.

Вважаємо, що вхідними даними задачі є вже згрупована інформація: сформовані групи виборців з однаковими у кожній групі рангами переваг. Проте допускається і занесення інформації кожним виборцем окремо.

# Опис алгоритму

У даному розділі наводяться алгоритми для знаходження переможців виборів.

Для визначення переможців Борда та Копленда скористаємося безпосередньо наведеними вище правилами, тобто реалізуємо їх програмно.

Складність алгоритмів, описаних нижче, прямо пропорційна кількості груп виборців та кількості кандидатів, що ще раз підтверджує приналежність даної задачі до Р-типу.

## Визначення переможця Борда

Для знаходження оцінок Борда кандидати ранджуються, тобто за кожне місце у шкалі виборців кандидат отримує певну кількість балів. Далі ці бали сумуються.

Введемо наступні змінні.

Нехай М – кількість кандидатів;

S – кількість груп виборців;

Nаme[M] – масив імен виборців;

Rаng[1..M, 1..S] – профіль переваг;

Many[S] – кількість виборців у кожній групі;

Bord[M] – масив оцінок кандидатів.

Розглядаємо окремо кожну групу виборців. Для цієї групи кандидат отримує оцінку [кількість виборців many[i]]\*([кількість кандидатів M] – [поточне значення лічильника j]). Знайдена оцінка додається до попередньої. Алгоритм продовжує роботу до тих пір, поки не будуть розглянуті усі групи виборців (i=S).

За правилом Борда отримуємо наступний алгоритм для знаходження оцінок Борда.



Рис. 4.1 Алгоритм знаходження оцінок Борда

## знаходження оцінки Копленда

Для знаходження оцінки Копленда крім вище наведених використаємо наступні змінні:

Kopl[M] – масив оцінок Копленда;

Vybor1, vybor2 – допоміжні змінні; використовуються для перегляду імен кандидатів з масиву імен name.

Порівняння проходить наступним чином.

Змінній vybor1 присвоюємо значення імені першого кандидата з множини імен name (contrl=1), а vybor2 – наступне (k=contrl+1). Якщо vybor1 знаходиться вище, ніж vybor2, у перевагах виборців усіх груп, то до оцінки Копленда (kopl[contrl]) кандидата vybor1 додається очко, а vybor2 (kopl[k]) – віднімається і навпаки. Далі змінній vybor2 присвоюється наступне значення з масиву імен (k=k+1), і процедура порівняння знову повторюється. Цикл продовжується до тих пір, поки не вичерпаються імена у списку кандидатів.

Після цього змінній vybor1 присвоюється друге ім’я із списку кандидатів (contrl=contrl+1), а vybor2 – третя. Знову проходить цикл по змінній vybor2. Цикл по змінній vybor1 закінчується тоді, коли буде переглянуто усіх кандидатів.

Отримуємо наступний алгоритм знаходження оцінки Копленда.



Рис. 4.2 Алгоритм знаходження оцінок Копленда (початок)



Рис. 4.3 Алгоритм знаходження оцінок Копленда (закінчення)

## Алгоритм визначення переможця за правилами Борда чи Копленда

Як можна було побачити, переможцем за Борда (Коплендом) є кандидат з найвищою сумою очок. Тому процес його визначення є однаковим для обох випадків , і я його об’єднала одним алгоритмом.

Як відомо, правила Копленда і Борда породжують множину розв’язків.

Для знаходження переможця використаємо наступні змінні:

K – кількість переможців;

Max – оцінка переможців;

Hl[M] – масив номерів переможців.

Спочатку масив номерів переможців заповнюємо нулями.

Масиви оцінок Копленда і Борда (kopl i bord) замінимо формальним масивом v[M].

Після того, як ми знайшли оцінки для кандидатів (масиви kopl i bord), серед них шукаємо максимальну (max). Далі відбираємо кандидатів, оцінка яких дорівнює max (їх номери з масиву name заносяться у масив hl), і рахуємо кількість переможців (k).

Переглядаємо заповнений масив hl. Якщо у ньому тільки перше значення є відмінне від нуля, то це означає, що ми знайшли єдиного переможця, і, отже, зберегли умову нейтральності.

В іншому випадку проглядаємо ті значення імен кандидатів (масив name), номери яких зустрічаються у масиві hl. Відібрані імена кандидатів впорядковуємо за списком. Переможцем стає той, ім’я якого знаходиться першим у даному списку.

У даному випадку умова нейтральності не зберігається.

Отже, ми отримали наступний алгоритм.



Рис. 4.4 Алгоритм визначення переможця (початок)



Рис. 4.5. Алгоритм визначення переможця (закінчення)

# Опис програми

## Вибір технології програмування

Найбільш поширеними парадигмами програмування, які до того ж можуть бути використані у даній курсовій роботі, є парадигми процедурного та об’єктно-орієнтованого програмування.

Парадигма процедурного програмування найбільше широко поширена серед існуючих мов програмування (наприклад, Сі і Паскаль). Тут явно виділяють два види різноманітних сутностей:

1) процедури, що виконують активну роль, тобто являються тим, що задає поведінку (функціонування) програми;

2) дані, що виконують пасивну роль, тобто являють тим, що опрацьовується засобом, продиктованим процедурами. Спроможність складати процедури з команд (операторів) і викликати їх є ключем даної парадигми. Особливістю цієї парадигми є "бічні ефекти", що виникають у тих випадках, коли різноманітні процедури, що використовують загальні дані, незалежно їх змінюють.

У процедурній парадигмі активна роль в організації поведінки приділяється процедурам, а не даним; причому процедура активізується викликом. Подібні засоби завдання поведінки зручні для описів детермінованої послідовності дій або одного процесу, або декількох, але строго взаємозалежних процесів.

При використанні програмування, орієнтованого на дані, активну роль відіграють дані, а не процедури. Тут із структурами активних даних зв'язують деякі дії (процедури), що активізуються тоді, коли здійснюється доступ до цим даним. Деякі спеціалісти називають цей засіб активації дій "демонами". Програмування, орієнтоване на дані, дозволяє організувати поведінку мало залежних процесів, що важко реалізувати в процедурній парадигмі. Мала залежність процесів означає, що вони можуть розглядатися і програмуватися окремо. Проте при використанні парадигми, керованої даними, ці незалежно запрограмовані процеси можуть взаємодіяти між собою без їхньої зміни (тобто без перепрограмування).

Парадигма об'єктного програмування на відміну від процедурної парадигми не розділяє програму на процедури і дані. Тут програма організується навколо сутностей, названих об'єктами, що включають локальні процедури (методи) і локальні дані (змінні). Поведінка (функціонування) у цій парадигмі організується шляхом пересилки повідомлень між об'єктами. Об'єкт, отримавши повідомлення, здійснює його локальну інтепретацію, базуючись на локальних процедурах і даних. Такий підхід дозволяє описувати складні системи найбільш природним чином. Особливо він зручний для інтегрованих ІС.

Об'єктно-орієнтоване програмування (ООП) базується на абстракціях даних. В основі цього методу лежить представлення предметної області у вигляді сукупності об’єктів, які взаємодіють між собою через передачу повідомлень. Модель абстракції даних полягає в інкапсуляції даних та операцій над ними в окремий класовий тип. Доступ до даних можлива лише за допомогою інкапсульованих операцій. Класовий тип є автономно завершеним і дозволяє повне або часткове наслідування для інших класів. ООП концентрується на суті задачі, для якої розробляється програма. Задача будується як сукупність об'єктів, які взаємодіють між собою за допомогою повідомлень. Елементи програми розробляються у відповідності з об'єктами в описі задачі. Суть ООП полягає у визначенні найбільш вдалих об'єктових типів. Об'єктовий тип служить модулем, який може бути використаним для розв'язку інших подібних задач. Такий підхід не потребує спеціальної обчислювальної техніки, але суттєво відрізняється характером мислення виконавців у порівнянні з процедурними мовами програмування.

Через очевидну простоту алгоритму для реалізації задачі найкраще вибрати процедурну парадигму програмування і мови Паскаль чи Сі. Звичайно, для написання гарного інтерфейсу можна взяти об’єктно-орієнтовані мови C Builder чи Delphi. Проте, як можна було побачити з розгляду алгоритму задачі, побудова інтерфейсу зводилася б до послідовного виведення вікон. Ще одним аргументом на користь мов Паскаль чи Сі є розміри програми, які б при використанні C Builder чи Delphi були набагото більшими.

Серед двох мов – Паскаль і Сі, – я оберу Паскаль як більш звичну для себе.

## Структура програми

Структурно дану програму можна поділити на блоки. Кожен блок може бути віднесений до однієї з функціональних груп:

1. Побудова інтерфейсу;
2. Реалізація алгоритмів, представлених у розділі 4.

Отже, програма має наступну структуру:



Рис. 5.1 Структура програми

Процедура victory – це реалізація алгоритму визначення переможця, описаного у попередньому розділі. Під час виклику даної процедури задається масив оцінок Борда чи Копленда, а також текст для виведення результатів (ним служать слова “Копленда” та “Борда”). У попередньому розділі вже було обгрунтовано, чому для визначення переможця за різними правилами використано єдиний алгоритм.

Процедура help виводить список імен кандидатів у нижньому рядку екрану. Вона введена для полегшення вводу інформації користувачем.

Процедура example містить дані контрольного прикладу. Вона введена для полегшення перевірки правильності роботи програми і носить демонстраційний характер.

Процедура right призначена для перевірки правильності вводу символу. Вона використовується при виборі внесення інформації (демонстрація контрольного прикладу чи самостійне внесення профілю) і виборі способу занесення даних (окремими виборцями чи працівниками виборчого комітету).

Перейдемо до розгляду основної програми.

Перш за все у ній проходить виклик і взаємозв’язок описаних вище процедур.

Процедури побудови інтерфейсу викликаються на початку роботи програми. Вони призначені для полегшення внесення даних.

Процедура victory визначає переможця і виводить результат голосування за певним правилом. Тому її виклик відбувається вкінці основної програми.

Опишемо змінні, які використовуються в основній програмі.

N: к-ть виборців;

M: к-ть кандидатів;

s: к-ть груп;

rang: профіль переваг;

a,b: для визначення оцінки Копленда (використовується у бінарних порівняннях);

kopl: масив оцінок Копленда;

vybor1, vybor2: змінні зовнішніх циклів при визначенні оцінки Копленда;

bord: масив оцінок Борда;

name: масив імен кандидатів;

k, i, j, l, r: допоміжні змінні;

many: масив груп виборців.

Опишемо структуру програми.

Спочатку програма просить внести усю потрібну інформацію. Користувач повинен вказати кількість виборців та кандидатів і спосіб занесення інформації (чи свої переваги буде вносити кожен виборець окремо, чи це буде проводити комісія, оперуючи згрупованими даними). Далі йде внесення профілю переваг і перевірка на його коректність. Профіль є некоректним, якщо у ньому зустрічається ім’я особи, яка не є вище вказаним кандидатом, або ж імена кандидатів повторюються.

У програмі використовуються алгоритми правил Борда та Копленда, вказані у попередньому розділі. Згідно отриманих оцінок визначається переможець за допомогою процедури victory, і проходить виведення результату.

Слід зауважити, що отримані переможці Копленда та Борда можуть не співпадати, що ще раз свідчить про недосконалість правил голосування більшістю голосів. Результати роботи алгоритму будуть показані у відповідному розділі.

Складність програми (у даному випадку розуміється час, затрачений на виконання), прямо пропорційно залежить від величини кількості виборців та кандидатів.

Так як дана програма носить більш демонстраційний характер, то я ввела межу для кількості виборців і кандидатів для того, щоб обмежити час виконання – 200 та 50 відповідно. В загальному воно не є суттєвим, так як завжди можна розбити виборчий округ на менший з умовою того, щоб виконувалось дане обмеження.

## Інструкція користувачеві

Дана програма призначена для визначення переможця виборів за правилами Копленда і Борда і порівняння отриманих результатів.

На початку роботи програми користувач може вибрати, чи проглядати результати розв’язку контрольного прикладу, чи вносити власні дані. В обох випадках визначаються переможці за Коплендом і Борда.

Спочатку працівники виборчого органу вносять загальну інформацію: кількість виборців у даному окрузі та кількість кандидатів. Далі заносяться імена кандидатів і вказується спосіб занесення профілів переваг: кожним виборцем окремо чи працівниками виборчого округу. В останньому випадку інформація є згрупована (аналогічно до контрольного прикладу).

Внизу екрана виводяться імена усіх кандидатів. Кожен виборець (чи працівник виборчого округу) вносить профіль переваг, розташовуючи кандидатів у строго визначеному порядку.

Для кожного виборця не допускається випадків байдужості; крім того, кандидати повинні бути строго ранджовані (тобто кожен з них займає своє місце у перевазі виборця, і на одному рівні не можуть знаходитись два кандидати). Імена кандидатів, які заносяться виборцями, повинні співпадати з іменами, вказаними на початку заповнення інформації.

Після занесення усіх цих даних видається результат роботи програми.

Спочатку виводиться переможець Копленда і вказується, чи він визначений із збереженням нейтральності. Для переможця вказується його оцінка. У противному випадку виводиться множина переможців (кандидатів, сума очок яких дорівнює максимальній оцінці).

Аналогічно визначається переможець Борда.

Як буде показано у контрольному прикладі, оцінки кандидатів, отриманих за правилами Борда і Копленда, можуть ранджуватись у протилежному порядку.

# Контрольний приклад

Нехай дано наступний профіль для 9 виборців і 5-ти кандидатів:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 4 | 1 | 3 |
| a  b  c  d  e | c  d  b  e  a | e  a  d  b  c | e  a  b  d  c |

У кожному стовпці кандидати розташовані у порядку зменшення їх значущості для кожної групи виборців. Тобто, для першого стовпця можна визначити переваги наступним чином: група виборців, що складається з однієї особи, вважає кандидата *a* найкращим. На другому місці вони ставлять кандидата *b*, на третьому місці *c* і т.д.

Продемонструємо розв’язок контрольного прикладу за правилом Копленда. Визначаємо оцінку Копленда.

Кандидат *a* є кращим за *b* для 1+1+3 виборців, а для 4-х виборців кандидат *b* є кращим за *a*. Визначимо такі переваги для кожного кандидата, порівняємо його з усіма іншими.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ab – 5  ac – 5  ad – 5  ae – 1 | ba – 4  ca – 4  da – 4  ea – 8 | bc – 5  bd – 4  de – 5 | cb – 4  db – 5  eb – 4 | cd – 5  ce – 5 | dc – 4  ec – 4 | de – 5 | ed – 4 |

Визначимо оцінку Копленда для кожного кандидата. Кандидат *a* є кращим за *b* (додаємо +1); він також є кращим за *c* та *d* (додаємо два рази +1) і гіршим за *e* (додаємо –1). Отже, оцінка Копленда для *a* рівна 2.

Знайдемо оцінку для інших кандидатів.

a=+1+1+1-1=2

b=-1+1-1+1=0

c=-1-1+1+=0

d=-1+1-1+1=0

e=+1-1-1-1=-2

Серед отриманих оцінок визначаємо максимальну. Як бачимо, вона дорівнює 2 і належить кандидату *a*. Отже, *a* – переможець Копленда.

Якби у нас отрималось два кандидати з максимальною оцінкою, наприклад *b* та *f*, ми б обрали кандидата *b*, так як він розташований ближче за алфавітом.

Для цього ж профілю знайдемо переможця Борда.

Отже, отримуємо такі оцінки:

a=1\*4+4\*0+1\*3+3\*3=16

b=3\*1+2\*4+1\*1+2\*3=18

c=2\*1+4\*4+0+0=18

d=1\*1+4\*3+2\*1+1\*3=18

e=1\*4+1\*4+3\*4+0=20

Переможцем за Борда є кандидат *е*.

Як бачимо, оцінки Борда ранджують кандидатів у порядку, протилежному до того, який отримується за оцінками Копленда.

ВИСНОВКИ

Дана курсова робота була присвячена огляду методів голосування більшістю голосів. Було проведено порівняльну характеристику кожного з методів і з їх множини обрано найкращі. До них відносяться

1. заможні за Кондорсе правила Копленла і Сімпсона, дерево багатоетапного виключення;
2. один з методів підрахунку очок – правило Борда.

Всі ці правила задовольняють умовам оптимальності за Парето, монотонності та анонімності. Крім того, правило Борда задовольняє також аксіомі участі та поповнення.

Для програмної реалізації було обрано методи Копленда і Борда.

Результати роботи програми продемонстровано на контрольному прикладі.

СПИСОК ЛІТЕРАТУРИ

1. Мулен Э. “Кооперативное принятие решений: Аксиомы и модели” – Москва, Мир, 1991.
2. Миркин Б. Проблема групового выбора. – Москва, Наука, 1974.

ДОДАТКИ

## Програма

uses wincrt;

label y, z;

type mas = string[6];

type ball =array[1..10] of shortint;

var N: byte; {к-ть виборцiв}

M: byte; {к-ть кандидатiв}

s: byte; {к-ть груп}

rang: array[1..10,1..100] of mas; {профiль переваг}

k,i,j,l,r,contrl: byte;

a,b: byte; {для проведення парних порiвнянь}

kopl: ball; {масив оцiнок Копленда}

vybor1, vybor2: mas;

bord: ball; {масив оцiнок Борда}

name: array[1..10] of mas; {масив iмен кандидатiв}

many: array[1..100] of byte; {масив груп виборцiв}

n1: array[1..10] of mas;

c: char;

{дані контрольного прикладу}

{---------------------------}

procedure example;

var i, j: byte;

begin

clrscr; M:=5; n:=9; s:=4;

name[1]:='a'; name[2]:='b'; name[3]:='c'; name[4]:='d'; name[5]:='e';

many[1]:=1; many[2]:=4; many[3]:=1; many[4]:=3;

rang[1,1]:='a'; rang[1,2]:='c'; rang[1,3]:='e'; rang[1,4]:='e';

rang[2,1]:='b'; rang[2,2]:='d'; rang[2,3]:='a'; rang[2,4]:='a';

rang[3,1]:='c'; rang[3,2]:='b'; rang[3,3]:='d'; rang[3,4]:='b';

rang[4,1]:='d'; rang[4,2]:='e'; rang[4,3]:='b'; rang[4,4]:='d';

rang[5,1]:='e'; rang[5,2]:='a'; rang[5,3]:='c'; rang[5,4]:='c';

gotoXY(15,1);

writeln('Завдання контрольного прикладу');

writeln; writeln('Число виборців: ', N);

writeln('Число кандидатів: ', M);

writeln('Профіль переваг:');

for i:=1 to 40 do

write('-');

writeln; write('Число виборців ');

gotoXY(19,7);

for i:=1 to s do

write(many[i], ' ');

writeln; gotoXY(19,9);

for i:=1 to M do

begin

for j:=1 to s do

write(rang[i,j], ' ');

gotoXY(19, 9+i);

end;

gotoXY(1,15);

end;

{---------------------------}

{перевіряє правильність вводу варіанту вибору}

procedure right;

label l;

begin

l: readln(c);

if (c<>'0') and (c<>'1') then

begin

write('Повторіть спробу: ');

goto l;

end;

end;

{---------------------------}

{виводить список імен кандидатів}

procedure help;

var x,y,i: byte;

begin

x:=WhereX;

y:=WhereY;

gotoXY(1,24);

write('Імена кандидатів: ');

for i:=1 to M do

if i<>M then write(name[i], ', ')

else write(name[i]);

gotoXY(x,y);

end;

{---------------------------}

{визначення переможця виборів}

procedure victory(v: ball; s: string);

var max, t: shortint;

hl: array[1..10] of byte;

begin

{визначення максимальної оцiнки}

help;

max:=v[1];

for i:=1 to M do

if max<v[i] then

max:=v[i];

t:=1;

{визначення кандидатiв з максимальною оцiнкою}

for i:=1 to M do

if (v[i]-max)=0 then

begin

hl[t]:=i;

t:=t+1;

end;

if (t-1)=1 then

begin

write('Переможець за ', s, ' iз збереженням нейтральностi: ');

writeln(name[hl[1]]); writeln('Сума очок - ', max);

end

else

begin

vybor1:=name[hl[1]];

for i:=2 to t-1 do

if name[hl[i]]<vybor1 then

vybor1:=name[hl[i]];

write('Переможець за ', s, ' без збереження нейтральностi: ');

writeln(vybor1); writeln('Сума очок - ', max);

writeln('обраний з множини найкращих:');

for i:=1 to t-1 do

writeln(name[hl[i]]);

end;

end;

{---------------------------}

{основна програма}

begin

gotoXY(21,1); writeln('Визначення переможця виборів');

writeln; writeln('Запуск контрольного прикладу - 1; Самостійне внесення профілю - 0');

right;

if c='1' then

begin

example;

help;

goto z;

end

else clrscr;

write('Введiть кiлькiсть кандидатiв: ');

readln(M);

write('Введiть кiлькiсть виборцiв: ');

readln(N);

writeln('Введiть iмена кандидатiв');

for i:=1 to M do

begin

write('Кандидат ', i, ': ');

readln(name[i]);

end;

writeln('Як буде здiйснюватись занесення iнформацiї?');

write('1- окремими виборцями, 0- комiтетом: ');

right;

if c='1' then

for i:=1 to N do

many[i]:=1;

clrscr; writeln('Введiть профiль переваг');

s:=1; contrl:=0;

while contrl<>N do

begin

if c='1' then writeln('Виборець ', s)

else writeln('Група ', s);

for i:=1 to m do

n1[i]:='';

help;

for j:=1 to M do

begin

y:readln(vybor1);

{перевірка на коректність введеного профілю}

r:=0; a:=0; b:=0;

n1[j]:=vybor1;

for l:=1 to M do

begin

if vybor1=name[l] then

begin

b:=1;

for a:=1 to M do

{таке ім'я вже було введено у даному профілі}

if (vybor1=n1[a]) and ((a-j)<>0) then r:=1;

end;

{ім'я введеного кандидата не співпадає із жодним із списку}

if (vybor1<>name[l]) and (l=M) and (b<>1)then r:=1;

end;

if r=1 then

begin

n1[j]:='';

writeln('Уважно вводьте імена кандидатів');

goto y;

end

else rang[j,s]:=vybor1; {профіль коректний}

end;

if c='0' then

begin

writeln('Кiлькiсть виборцiв у групi ', s);

readln(many[s]);

contrl:=contrl+many[s];

end

else

contrl:=contrl+1;

s:=s+1;

clrscr;

end; {while}

{Визначення оцiнок Копленда}

z: contrl:=1;

while contrl<=M do

begin

k:=contrl+1;

vybor1:=name[contrl]; vybor2:=name[k];

while k<=M do

begin

i:=1; a:=0; b:=0;

while i<=s do

begin

for j:=1 to M do

if rang[j,i]=vybor1 then l:=j

else

if rang[j,i]=vybor2 then r:=j;

if l<r then a:=a+many[i]

else

if l>r then b:=b+many[i];

i:=i+1;

end;

if a>b then

begin

kopl[contrl]:=kopl[contrl]+1;

kopl[k]:=kopl[k]-1;

end

else

if a<b then

begin

kopl[k]:=kopl[k]+1;

kopl[contrl]:=kopl[contrl]-1;

end;

k:=k+1;

vybor2:=name[k];

end; {while по k}

contrl:=contrl+1;

end; {while по contrl}

{визначення оцiнок Борда}

for i:=1 to s do

for j:=1 to M do

begin

for k:=1 to M do

if rang[j,i]=name[k] then r:=k;

bord[r]:=many[i]\*(M-j)+bord[r];

end;

victory(kopl, 'Коплендом');

writeln ('Натисніть будь-яку клавішу ');

readkey; writeln;

victory(bord, 'Борда');

end.Результати роботи програми

### Самостійне внесення профілю

*Введіть кількість кандидатів: 5*

*Введіть кількість виборців: 9*

*Введіть імена кандидатів*

*Кандидат 1: а*

*Кандидат 2: b*

*Кандидат 3: c*

*Кандидат 4: d*

*Кандидат 5: е*

*Як буде здійснюватись занесення інформації?*

*1-окремими виборцями, 0 – комітетом: 0*

*Введіть профіль переваг*

*Група 1*

*a*

*b*

*c*

*d*

*e*

*Кількість виборців у групі 1: 1*

*Група 2*

*c*

*d*

*b*

*e*

*a*

*Кількість виборців у групі 2: 4*

*Група 3*

*e*

*a*

*d*

*b*

*c*

*Кількість виборців у групі 3: 1*

*Група 4*

*e*

*a*

*b*

*d*

*c*

*Кількість виборців у групі 4: 3*

*Переможець за Коплендом із збереженням нейтральності – а*

*Сума очок – 2*

*Переможець за Борда із збереженням нейтральності – е*

*Сума очок – 20*

### Демонстрація роботи контрольного прикладу

