Дніпровський ліцей інформаційних технологій при  
Дніпропетровському національному університеті

Кафедра Інформатики

**КУРСОВА РОБОТА**

**Тема:** Розв'язування задачі про призначення повним перебором та деякі модифікації цієї задачі

**Виконав:**

Ліцеїст 10-Г класу

*Олешко Олександр Олегович*

**Керівники:**

*Ентін Йосиф Абрамович, вчитель-методист, к. ф.-м. н.*

Дніпро

2017

ЗМІСТ

**ВСТУП** ................................................................................................ 3

Тема роботи ……………………………….…………………... 3

Актуальність роботи ………………………….………………. 3

Мета роботи ……………………….…………………………... 3

Задачі роботи …………………….……………………………. 4

**ОСНОВНА ЧАСТИНА** .................................................................... 5

1. ТЕОРЕТИЧНА ЧАСТИНА …..….….................................... 5
2. ОПИС РОБОТИ ..................................................................... 5
3. БЛОК-СХЕМА ..................................................................... 13
4. НАПРЯМКИ, ДЕ МОЖНА ВИКОРИСТАТИ РОБОТУ... 13
5. ПРОГРАМНІ ТА АПАРАТНІ ВИМОГИ .......................... 14
6. ВИКОРИСТАНІ ПРОГРАМИ ............................................ 14
7. КОМПЛЕКТАЦІЯ ПРОГРАМИ…...…...………………... 15

**ВИСНОВКИ** ..................................................................................... 16

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ** .................................... 17

**ДОДАТОК 1**………………………………………………………...17

**ДОДАТОК 2**………………………………………………………...19

Вступ

***Тема роботи***

**Задача про призначення:** "кращий працівник для виконання даної роботи" - ось підходяще короткий опис задачі про призначення. У цьому завданні необхідно призначити працівників на певні роботи; кожен працівник може виконувати будь яку роботу, хоча і з різним ступенем майстерності. Якщо на деяку роботу призначається працівник саме тієї кваліфікації, яка необхідна для її виконання, тоді вартість виконання роботи буде нижче, ніж при назначені на дану роботу працівника невідповідної кваліфікації. Мета завдання - знайти оптимальне (мінімальної вартості) розподіл працівників повсім заявленим роботам.

***Актуальність роботи***

Дана робота присвячена такій темі, як «задачі про призначення». Ця тема є важливою у наш час, адже завжди потрібно знайти кращого працівника для виконання даної роботи. Тобто найшвидше виконання роботи – це завжди краще, а у деяких випадках необхідно так розкидати роботи, щоб встигнути вчасно чи вкластися у деякий план.

***Мета роботи***

Метою роботи є знаходження найшвидшого варіанту виконання робот, коли одну роботу виконує один працівник, а також розгляд деяких окремих випадків, а саме: виконання усіх робот одним працівником.

***Задачі роботи***

У роботі розглянуто декілька задач, пов’язаних із ефективністю алгоритмів сортування в певних умовах.

1. Методом повного перебору знайти найшвидший варіант виконання робіт за мінімальний час, при чому, коли один працівник виконує одну роботу.
2. Розглянути окремі випадки:
   1. Один працівник виконує усі роботи. Чи вигідніше це, ніж виконання робіт іншими працівниками.
   2. Якщо ми маємо не час, за який виконується робота, а ціна. Знайти найменшу ціну.

Основна частина

1. ***Теоретична частина***
   1. **Визначення терміну «повного перебору»**

Метод «грубої сили» (або повний перебір) — метод рішення криптографічної задачі шляхом перебору всіх можливих варіантів ключа. Складність повного перебору залежить від кількості всіх можливих рішень задачі. Якщо простір рішень дуже великий, то повний перебір може не дати результатів протягом декількох років або навіть століть.

* 1. **Оцінка ефективності алгоритму повного перебору**

Алгоритм повного перебору є найпростішим алгоритмом задля знайдення найкращої відповіді для задачі, але він малоефективний, адже він перебирає усі можливі варіанти задачі, не відсікаючи нічого. Тобто при n робітниках ми маємо n! Варіантів виконання роботи, і для трьох робітників це буде 6, а для шести робітників вже 720.

1. ***Опис роботи***
   1. **Елементи програми**

Робота виконана у програмному середовищі Microsoft Visual Studio 2015 Community мовою C# .NET. Для розробки дизайну використовувалась технологія Windows Forms. Використані такі елементи управління.

* Label, TextBox, Button, RichTextBox – для взаємодії з користувачем і відображення текстової інформації та різного походження(статичної та змінюваної програмно)
* dataGridView – для відображення отриманих даних та візуалізації
* MessageBox – для виведення повідомлень про помилку або допоміжної інформації

Проект складається з чотирьох розділів, два з яких відповідають задачам, які було за потрібним розглянути у роботі («Повний перебір», «Окремі випадки») та двох допоміжних розділів, що допомагають краще зрозуміти поданий матеріал та розібратися як виконується знаходження відповідей («Теорія та інструкція» та «Приклади»).

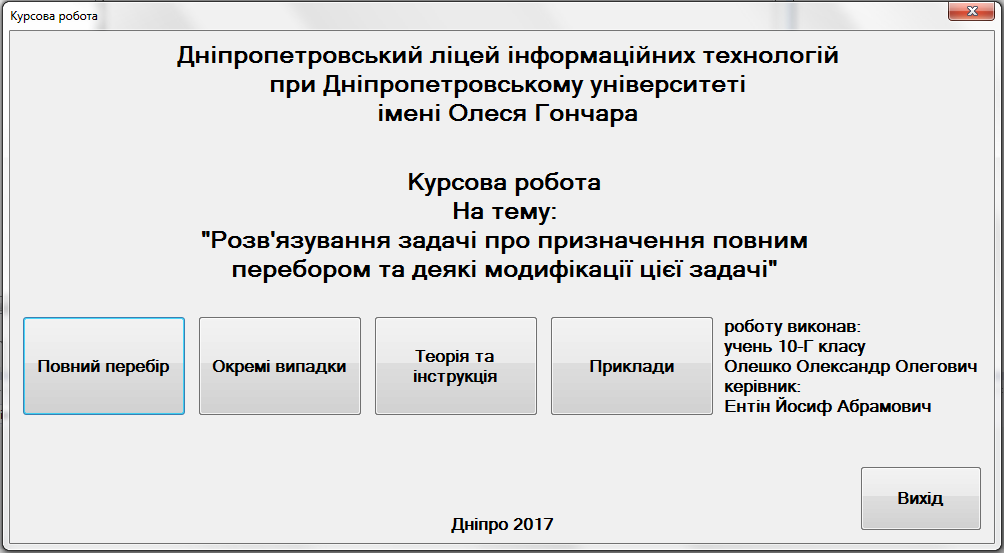
* 1. **Основні розділи програми**
     1. *Головне меню*

Рис. 2.1. Головне меню

При запуску програми відкривається головне меню(Рис. 2.1.), з якого можна перейти до одного з п’яти розділів, або вийти з програми. Повернення до головного меню доступне з будь-якого розділу і трапляється при натисканні на таку кнопку(Рис. 2.2.).

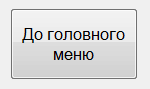
Якщо введено невірні дані до текстових полів, програма виведе повідомлення про помилку у MessageBox, де буде вказана причина помилки. Також MessageBox з’являється після натискання Button’a «Побудувати матрицю», щоб зрозуміти як її заповнити(Рис. 2.3.).

Рис. 2.2. Кнопка повернення до головного меню

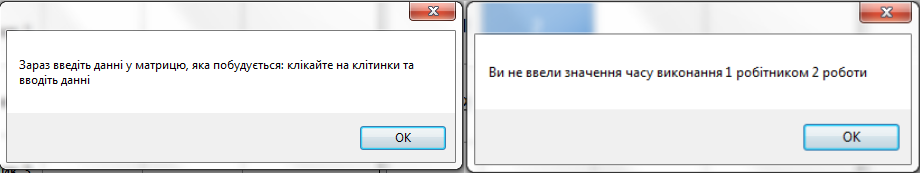


Рис. 2.3. Повідомлення про помилку

Розглянемо детальніше особливості інтерфейсу основних розділів.

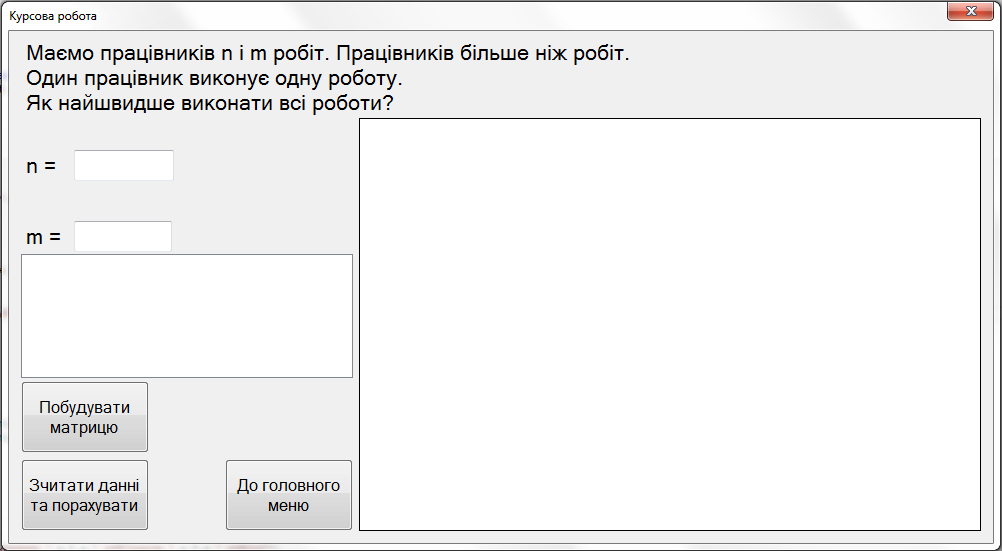
* + 1. *Повний перебір*

Рис. 2.4. Інтерфейс розділу «Повний перебір»

У даному розділі розглянуто знаходження найшвидшого часу виконання робіт, якщо один працівник має право виконувати лише одну роботу. З елементів textBox буде зчитано кількість робітників та робіт, якщо кількість робіт буде перевищувати кількість робітників (що за умовою не є можливим), то буде виведено MessageBox з посиланням на те, що введено некоректні данні.

Після того, як будуть введені данні і було натиснута кнопка «Побудувати матрицю» і умови будуть виконані, то буде виведено MessageBox з посиланням на те, що зараз потрібно заповнити матрицю (клікаючи на комірки та таким чином заповнюючи їх).

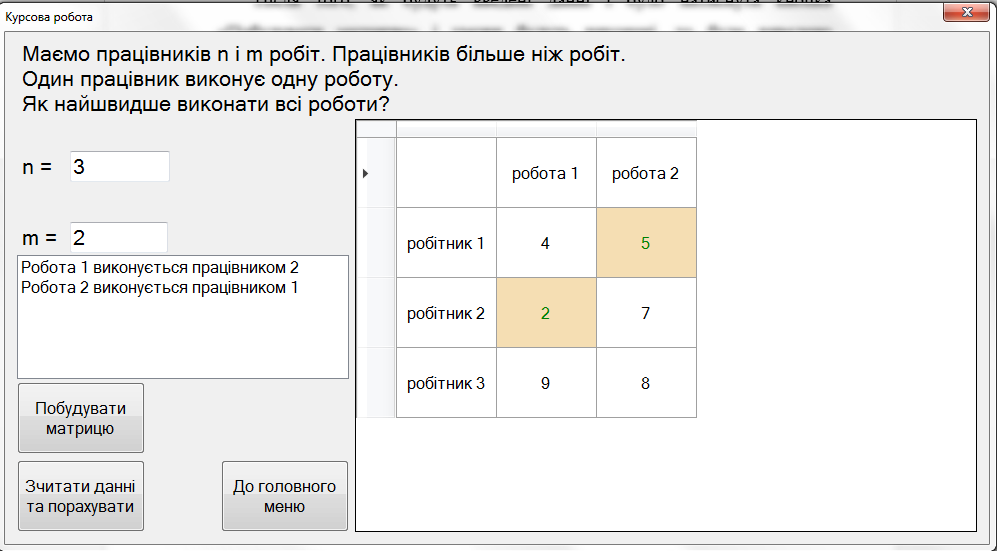
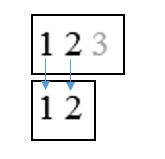
Після того, як буде введено усі данні та натиснута кнопка «Зчитати данні та порахувати», буде виведено відповіді у listBox та правильні відповіді будуть виділені у dataGridView. Саме це і будуть наші відповіді.(Рис. 2.5.).

Рис 2.5. Приклад вирішення задачі про призначення

Розглянемо вирішення цієї задачі:

Нехай у нас є n робітників і m робіт. Тоді у нас створюється два масиви. А саме: з n і m елементів. Вони виглядають так (якщо n=3, m=2):

**An[1; 2; 3], Am[1; 2]**

 Тобто, це нумерації робітників та робіт, саме так я порівнював. Було узято перші m елементів масиву **An** і підставлялося під масив **Am** саме так відбувалися і інші порівняння. Підставлення виглядало так:

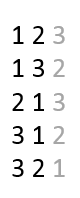
На малюнку ми бачимо, що перший робітник буде виконувати першу роботу, а другий другу. Але як будуть виглядати інші варіанти? Для цього я використав метод Найрани задля перебору усіх можливих перестановок чисел. Він виглядає так:

Крок 1: знаходимо найбільший j, для котрого aj<aJ+1:



Крок 2: збільшити aj. Для цього потрібно знайти найбільший l, для якого al>aj. Міняємо al і aj місцями:

 Крок 3: записати aj+1, …, an в зворотному порядку. У нашому випадку це нічого не змінюється.



Отже взагалі ми маємо n! варіантів. В нашому випадку (при n=3) це 6 варіантів, а саме:

Отже ми бачимо усі наші різні варіанти, коли який робітник буде виконувати першу, а коли другу.

Знаходиться найкращий варіант таким чином: ми дивимось який робітник виконує роботу найдовше і зберігаєм цей результат – найдовше виконання роботи. Нам потрібно знайти мінімальне найдовше виконання. Саме це і буде нашою відповіддю: та матриця, яка має мінімальний найдовший варіант.

*Інструкція до цього розділу програми:*

У цьому розділі зображено задачу, яка розв’язується методом повного перебору. Задля цього:

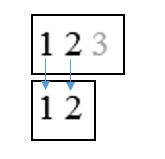
1. Введіть кількість робітників – n, та кількість робіт – m. Зверніть увагу, що n повинно бути більшим за m.
2. Натисніть кнопку «Побудувати матрицю».
3. Заповніть її. Подвійний клік на одну з комірок та заповніть її значенням. Наприклад: 5.
4. Коли ви заповнили усі комірки з на перехресті робітників та робіт, натисніть кнопку «Зчитати данні та порахувати».
5. Відповідь буде виділена та виведена списком.
   * 1. *Окремі випадки*

Тут розглянуто деякі окремі випадки для задачі про призначення, а саме:

* Найвигідніша ціна: завжди буде існувати обмеження у фінансах, саме для цього було розглянуто варіант з мінімальною ціною
* Один робітник на усі роботи: чи буде цей варіант вигідніше, замість того, щоб роздавати роботи між іншими працівниками
  + - 1. *Найвигідніша ціна*

Нехай у нас є n робітників і m робіт, але вони виконують роботи не за час, а за деякі ціни. Тоді у нас створюється два масиви. А саме: з n і m елементів. Вони виглядають так (якщо n=3, m=2):

**An[1; 2; 3], Am[1; 2]**

 Тобто, це нумерації робітників та робіт, саме так я порівнював. Було узято перші m елементів масиву **An** і підставлялося під масив **Am** саме так відбувалися і інші порівняння. Підставлення виглядало так:

На малюнку ми бачимо, що перший робітник буде виконувати першу роботу, а другий другу. Але як будуть виглядати інші варіанти? Для цього я використав метод Найрани задля перебору усіх можливих перестановок чисел. Він виглядає так:

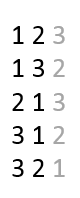
Крок 1: знаходимо найбільший j, для котрого aj<aJ+1:



Крок 2: збільшити aj. Для цього потрібно знайти найбільший l, для якого al>aj. Міняємо al і aj місцями:

Крок 3: записати aj+1, …, an в зворотному порядку. У нашому випадку це нічого не змінюється.





Отже взагалі ми маємо n! варіантів. В нашому випадку (при n=3) це 6 варіантів, а саме:

Отже ми бачимо усі наші різні варіанти, коли який робітник буде виконувати першу, а коли другу.

Знаходиться найкращий варіант ціни таким чином: ми знаходимо суму усіх цін, і знаходимо мінімальну суму. Саме це і буде нашою відповіддю.

*Інструкція до цього розділу програми:*

У цьому розділі зображено окремий випадок задачі, коли ми замінюємо час виконання роботи на ціну виконання роботи. Це також метод повного перебору, але я дещо змінив алгоритм знаходження відповіді: замість знаходження мінімального «максимального значення», ми знаходимо мінімальну суму. Задля цього:

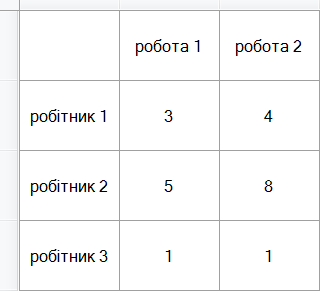
1. Введіть кількість робітників – n, та кількість робіт – m. Зверніть увагу, що n повинно бути більшим за m.
2. Натисніть кнопку «Побудувати матрицю».
3. Заповніть її. Подвійний клік на одну з комірок та заповніть її значенням. Наприклад: 5.
4. Коли ви заповнили усі комірки з на перехресті робітників та робіт, натисніть кнопку «Гроші».
5. Відповідь буде виділена та виведена знизу форми.
   * + 1. *Один робітник на всі роботи*

Рис 2.6. Початкова матриця для дальніх дій

Нехай у нас є n робітників і m робіт. Розглянемо випадкову матрицю, коли n=3, m=2:

Ми повинні визначитись: чи вигідніше віддавати усі роботи одному працівнику: чи швидше він виконає їх всі, ніж інші (Рис.2.6).

Крок 1: знайдемо мінімальний час виповнення першої роботи:

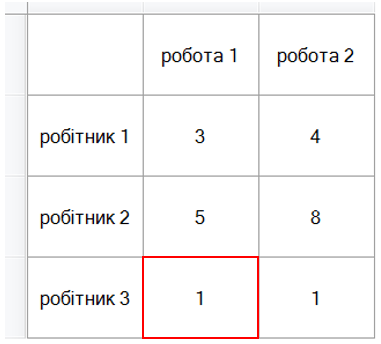
 Крок 2: знайдемо суму часу виконання усіх робіт тим робітником, у якого час виконання роботи 1 був мінімальним: у нашому випадку сумма дорівнює 2.

Рис 2.7. Матриця з мінімальним часом виконання першої роботи

Крок 3: порівнюємо цю суму з часом виконання робіт іншими робітниками. Якщо сума менша, або дорівнює усім іншим роботам, то це буде вигідніше віддати усі роботи одному працівнику, замість того, щоб розподіляти роботи між усіма.

*Інструкція до цього розділу програми:*

У цьому розділі зображено окремий випадок задачі, коли ми задаємося питанням: чи буде вигідніше віддати усі роботи комусь одному, замість роздачі їх усім. Задля цього:

1. Введіть кількість робітників – n, та кількість робіт – m. Зверніть увагу, що n повинно бути більшим за m.
2. Натисніть кнопку «Побудувати матрицю».
3. Заповніть її. Подвійний клік на одну з комірок та заповніть її значенням. Наприклад: 5.
4. Коли ви заповнили усі комірки з на перехресті робітників та робіт, натисніть кнопку «Один робітник».
5. Відповідь буде виділена та виведена знизу форми.
6. ***Блок-схема***
7. ***Напрямки, де можна використати роботу***

Мета створення роботи була: знайти найшвидший час виконання роботи. Адже є дуже багато ситуацій, у яких ми можемо застосувати «розумний» розподіл праці, що допоможе зберегти час або кошти. Також у роботи є повчальна частина, яка навчає використовувати метод «повний перебір», який є рішенням майже усіх задач про призначення.

1. ***Програмні та апаратні вимоги***

Апаратні вимоги.

* Частота процесора 1.3 GHz або вище
* 512 MB RAM
* 3 MB вільного місця на жорсткому диску
* Роздільна здатність монітору 1024 x 768 або вище

Підтримуючі операційні системи.

* Windows 10
* Windows 7, Windows 7 with SP1
* Windows 8, Windows 8 Pro, Windows 8.1
* Windows XP

1. ***Використані програми:***

* Visual Studio 2015 Community
* Microsoft Word 2010/2013
* Microsoft Excel 2013
* Google Chrome
* Paint
* GIMP 2
* Adobe Acrobat Pro
* Notepad++

1. ***Комплектація програми***

Для встановлення програми необхідно скопіювати папку «oleshko\_o\_kyrsovaya» у директорію на комп’ютер за власним розсудом. ЇЇ зміст такий.

1. oleshko\_o\_kyrsovaya.exe – файл запуску програми для подальшої роботи
2. Папка priklad з файлами: a0, a1, a2, де розібрано приклади знаходження відповідей на кожну серед задач.
3. Папка instr з файлами: a0, a00, a000, a001, a01, a1, де детально розглянуто інструкцію та теорію на кожен з видів задач.

**Висновки**

При виконанні роботи виникали такі проблеми, як знаходження алгоритму повного перебору варіантів перестановок чисел, адже існувало декілька таких варіантів, і був обраний найпоширеніший: лексографічний порядок, який реалізовувався методом Найрани. Також під час виконання роботи були вивчені та відкриті такі речі:

* Повний перебір може бути реалізованим декількома методами, так званими «порядками». Є лексографічний порядок, а є антилексографічний порядок. Відрізняються вони лише тим, що у першого відбувається перебір з останньої цифри, а у другого – з першої.
* Більшість варіацій «кращого часу» і «кращої ціни» на однакових даних співпадають, адже ми мали такі рамки, що один працівник виконує одну роботу.
* Для того, щоб почати шукати «одного робітника на всі роботи» не є потрібним перебирати усі суми робіт всіх працівників, лише потрібно знайти мінімальне виконання першої роботи, від чого і далі відштовхуватись.

Список використаних джерел

***Літературні джерела:***

1. Таха «Введення в дослідження операцій. 7 видавництво.».
2. Окулов «Програмування в алгоритмах».

***Інтернет-джерела:***

1. Інтернет-енциклопедія «Вікіпедія» <https://ru.wikipedia.org/>, <https://uk.wikipedia.org/>.
2. MSDN <https://msdn.microsoft.com/>.
3. Математика <http://ru.math.wikia.com/>.

**Додаток 1**

Алгоритм Найрани:

listBox1.Items.Clear();

for (i = 1; i <= n; i++)

{

for (j = 1; j <= m; j++)

{

if (dataGridView1.Rows[i].Cells[j].Value == null)

{

MessageBox.Show("Ви не ввели значення часу виконання " + i + " робітником " + j + " роботи");

break;

}

raba[i-1, j-1] = Convert.ToInt32(dataGridView1.Rows[i].Cells[j].Value);

}

}

string otv = "";

a = new int[n];

sumr = new int[m];

//Факторіал числа n - кількість варіантів.

for (i = 0; i < n; i++)

{

a[i] = i + 1;

}

for (i = 1; i <= n; i++)

{

nf = nf \* i;

}

//Перший раз перебираю і знаходжу максимальну суму рядка для варіанту при n = 4: 1 2 3 4

maxpr = -1;

for (dls = 0; dls < m; dls++)

{

sumr[dls] = raba[(a[dls] - 1), dls];

if (sumr[dls] > maxpr)

{

maxpr = sumr[dls];

}

}

if (maxs > maxpr)

{

otv = "";

maxs = maxpr;

for (int lon = 0; lon < m; lon++)

{

otv = otv + " " + a[lon].ToString();

}

}

//Тепер я використовую алгоритм Найрани. Короткий опис:

for (k = 0; k < nf; k++)//Крок 1: знайти найбільший j, для якого a [j] <a [j + 1].

{

l = 0;

j = 0;

for (v = n - 2; v >= 0; v--)

{

if (a[v] < a[v + 1] && v > j)

{

j = v;

}

}

for (z = j + 1; z < n; z++)//Крок 2: збільшити a [j]. Для цього потрібно знайти найбільше l, для якого a [l] > a [j]. Потім поміняти місцями a [j] і a [l].

{

if (a[j] < a[z] && z > l)

{

l = z;

}

}

swap = a[j];

a[j] = a[l];

a[l] = swap;

b = new int[n - (j + 1)];//Крок 3: записати a [j + 1], ..., a [n] в зворотному порядку.

r = 0;

for (x = j + 1; x < n; x++)

{

b[r] = a[x];

r++;

}

Array.Reverse(b);

r = 0;

for (x = j + 1; x < n; x++)

{

a[x] = b[r];

r++;

}

//Переглядаю інші варіанти перебору, як описано вище.

maxpr = -1;

for (dls = 0; dls < m; dls++)

{

sumr[dls] = raba[(a[dls] - 1), dls];

if (sumr[dls] > maxpr)

{

maxpr = sumr[dls];

}

}

if (maxs > maxpr)

{

otv = "";

maxs = maxpr;

for (int lon = 0; lon < m; lon++)

{

otv = otv + " " + a[lon].ToString();

}

}

}

otv = otv.Trim();

string[] otvet;

otvet = otv.Split(' ');

for (int otvi = 0; otvi < m; otvi++)

{

cvet = Convert.ToInt32(otvet[otvi]);

dataGridView1.Rows[cvet].Cells[otvi+1].Style.ForeColor = Color.Green;

dataGridView1.Rows[cvet].Cells[otvi+1].Style.BackColor = Color.Wheat;

listBox1.Items.Add("Робота " + (otvi + 1).ToString() + " виконується працівником " + otvet[otvi]);

}

**Додаток 2**

Виконання роботи одним працівником:

j = 0;

for (i = 0; i < n; i++)

{

if (raba[i, j] < mina)

{

mina = raba[i, j];//знаходження мінімального часу виконання першої роботи (записується індекс працівника)

mini = i;

}

}

for (j = 0; j < m; j++)

{

sumstr = sumstr + raba[mini, j];

}

for (i = 0; i < n; i++)

{

for (j = 0; j < m; j++)

{

if (raba[i, j] < sumstr)//рахується кількість робіт, що не перевищує суми "кращого виконавця" першої роботи

{

kolvo++;

}

}

}

if (kolvo == m)//якщо ця сумма дроівнює m, то робітник кращий, адже m - кількість робіт у нього і він виконує їх швидше у сумі за інших

{

label3.Text = "Робітник " + (mini + 1).ToString() + " буде виконувати усі роботи скоріше за інших.";

dataGridView1.Rows[mini +1].Cells[0].Style.ForeColor = Color.Green;

dataGridView1.Rows[mini + 1].Cells[0].Style.BackColor = Color.Wheat;

}

else

label3.Text = "Ніякий робітник не буде виконувати усі роботи скоріше за інших.";