третьому стартовому майданчику зіграє другий гравець з команди 1, другий гравець з команди 2 та перший гравець з команди 4. На четвертому стартовому майданчику зіграє другий гравець з команди 4, другий гравець з команди 3 та третій гравець з команди 1. На п'ятому стартовому майданчику зіграє третій гравець з команди 2, третій гравець з команди 4 та третій гравець з команди 3. Таким чином, ми отримали розклад турніру.

**Література**

1. Белоусов В. Д. Латинские квадраты, квазигруппы и их приложения. / В. Д. Белоусов, Г. Б. Белявская. – Кишинев: Штиница, 1989. – 78 с.

УДК 004:371.8:378

# ПРОБЛЕМА ОБРАННЯ МОВИ ПРОГРАМУВАННЯ, ЯК ІНСТРУМЕНТУ ДЛЯ НАВЧАННЯ ТА РОЗРОБКИ

***Ю. С. Антонов, К. Р. Дзігора***

Питання, яку мову програмування обрати та яка краща, дуже часто виникає як у тих, хто збирається вивчати програмування(вже вивчає), так і у тих, хто ці мови програмування викладає [1, 2]. Зрозуміло, що кожна сторона має свої критерії обрання тієї чи іншої мови програмування. Існують роботи [2], де пропонується обирати мову програмування в залежності від задачі та її предметної галузі, вимог до процесу розробки, швидкодії та потреби у ресурсах, тощо. Мета цієї роботи висвітлити деякі проблеми обрання мови програмування та показати, що кожна мова програмування є інструментом для розв'язання певного класу задач. Автори, у жодному випадку, не ставлятьза метудовести перевагу використання якоїсь мови програмування.

Бажаючих вивчати програмування можна поділити на дві категорії: перша – не знає жодної мови програмування, друга - знає хоча б одну мову програмування. Ті, хто вивчає вже не першу мову програмування, роблять це для кар'єрного зростання, самовдосконалення, підвищення конкурентоспроможності, тощо. Ті, хто не знає жодної мови програмування, фактично схожі на чисту дошку, на якій буде "писати" той чи інший наставник, і віднього залежить дуже багато. Серед усіх проблем, що виникають під час навчання, хотілося б виділити наступні: відсутність мотивації до навчання, недостатність знань (досвіду) та «євангелізм». Відсутність мотивації це проблема людей з першої категорії, напрям навчання яких часто обрано іншими людьми. Друга проблема стосується як викладачів так і слухачів: недостатньо знань або вмінь для опанування курсу (розв'язання задачі), вміння писати програми, але відсутність навичок доступно пояснити, як це робити, тощо.Найбільш загрозливою проблемою у наш час можна вважати «євангелістів» тієї чи іншої мови програмування(IT технології) та подібних до них людей. Такі люди, свідомо чи несвідомо, формують світогляди або стереотипи навколо тієї чи іншої технології. Лаври «універсальної»або «самої крутої» мови програмування вже отримували C та C++,Java таC#, PHP таPython. Якщо в один той самий час, дві компанії будуть проводити навчання або інші заходи, то ми можемо почути в одному таборі що «*мова X краща за мову Y*», а в іншому – навпаки *«Y значно крутіша ніж X*». Не кожна людина, яка відвідає обидва заходи зможе зробити правильні висновки. Наприклад, для задач з однієї предметної галузі підходить мова X, а з другої – Y, а для третьої – Z. У математичному аналізі одна й та сама теорема може мати два і більше доказів, під час розгляду яких кажуть, що, наприклад, перший доказ є коротким, але складним, а другий – легше сприймається, але вдвічі довший. Так само, при порівнянні алгоритмів, ми кажемо, що перший – найшвидший, а другий використовує

мінімальний обсяг пам'яті. В практиці одного з авторів був випадок, коли студент розв'язуючи проблему виявлення плагіату у роботах студентів, обрав у якості інструменту – PHP саме під впливом «євангелістів», хоча йому пропонувались C++, Java, C#. Наступного року, цей самий усвідомивши недоліки обраного інструменту, вдало реалізував програмний комплекс для цієї задачі на мові C#.

Розглянемо алгоритм, код якого на *C* буде виглядати наступним чином:

```
for (int i = 1; i <= M; i++)
    for (int j = 1; j <= M; j++)
        for (int k = 1; k <= M; k++)
            for (int t = 1; t <= M; t++)
                s=i+j+k+t;
```

Аналогічний код було створено на інших мовах програмування, а саме *C++*, *Fortran*, *Java*, *C#*, *PHP*,*Python*, *JavaScript*. Після цього відбувалось вимірювання часу потрібного для виконання. Відзначимо, що вимірювався час роботи виключно нашого алгоритму, а не програми в цілому. Запуск відбувався 5 разів і бралось найменше значення (див. табл. 1).

*Таблиця 1*

**Результати тестування**

| № | Мова | *M* | | | | | |
|---|---|---|---|---|---|---|---|
| | | **50** | **100** | **150** | **200** | **250** | **300** |
| 1 | Cgcc | 0,000 | 0,156 | 0,828 | 2,594 | 6,344 | 13,095 |
| 2 | Cgcc:x64 –O | 0,000 | 0,031 | 0,156 | 0,484 | 1,156 | 2,360 |
| 3 | C gcc:x64 -O2 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 4 | Fortran gfortran | 0,000 | 0,203 | 1,063 | 3,406 | 8,375 | 17,391 |
| 5 | Fortran gfortran -O | 0,000 | 0,031 | 0,156 | 0,484 | 1,172 | 2,406 |
| 6 | Javajavac | 0,000 | 0,046 | 0,172 | 0,500 | 1,187 | 2,406 |
| 7 | C# csc | 0,062 | 0,122 | 0,369 | 1,022 | 2,383 | 4,833 |
| 8 | C# csc/o | 0,060 | 0,092 | 0,220 | 0,555 | 1,240 | 2,469 |
| 9 | PHP php7 (i++) | 0,186 | 2,953 | 14,950 | 47,211 | 115,179 | 238,540 |
| 10 | PHP php7( ++i) | 0,170 | 2,716 | 13,637 | 43,020 | 105,211 | 218,852 |
| 11 | PHP php5 (i++) | 0,241 | 3,818 | 19,207 | 60,750 | 147,946 | 306,009 |
| 12 | PHP php5 (++i) | 0,219 | 3,494 | 17,487 | 55,205 | 134,789 | 279,996 |
| 13 | Pythonpython | 0,688 | 10,767 | 57,756 | 197,302 | 504,366 | 1114,116 |
| 14 | Python python(pyproc) | 0,464 | 7,186 | 35,551 | 121,888 | 319,627 | 713,882 |
| 15 | Pythonpypy | 0,015 | 0,265 | 1,266 | 3,938 | 9,485 | 19,517 |
| 16 | Python pypy (pyproc) | 0,016 | 0,141 | 0,719 | 2,432 | 6,198 | 11,266 |
| 17 | JavaScript nodeJS | 0,027 | 0,398 | 1,991 | 6,258 | 15,141 | 31,047 |

Мови *C* та *C++* показали однакові результати у тесті, тому у табл. 1 присутні результати лише для першої мови. Як видно з рядків 1-5, використання відповідного компілятору без оптимізації (рядки 1, 4) робить програму повільнішою в 5-5,5 разів порівняно з оптимізованими за допомогою опції*–O* (рядки 2, 5), а використання опцій *–O2* або *–O3* дозволяє компілятору «побачити», що накопичення у циклі не відбувається і тому він непотрібен. Для мов *C*, *C++*, *Java* ,*C#*,*JavaScript* різниці в швидкості між використанням *i++* та *++i* не було. Якщо взяти мову *C* з опцією *–O2* за еталон та порівняти її з мовами *Java* та *C#* з опцією /оми побачимо, що при *M=100* програми повільніші на 47% та 194% відповідно, то вже при *M=300* різниця становить лише 2% та 5% відповідно. Для мови *PHP* варіант з i++ працює на 9–10 % повільніше ніж варіант з ++i (рядки 9–12). Для мови *Python* використання замість інтерпретатора *pypy*, класичного *python* уповільнить роботу програми в 50 разів (рядки 13, 15), а якщо тестовий алгоритм винести поза межі окремої процедури, у глобальний блок, то програма уповільниться мінімум на 50 %. Слід зазначити, що наведені у таблиці дані дозволяють маніпулювати думкою, якщо відображати їх не повністю.

Підводячи підсумок хотілося б зазначити, що будь-яка мова програмування, це інструмент, що дозволяє розв'язувати певні класи задач. Якість розв'язання буде залежати від майстерності автора та вдало підібраного для цього інструменту.

**Література**

1. Шевчук П. Г. Основні підходи добору мови та середовища програмування як засобів навчання. [Електронний ресурс] / П. Г. Шевчук // Інформаційні технології і засоби навчання. – 2010. – № 3. – Режим доступу: http://www.nbuv.gov.ua/e-journals/ITZN/em6/emg.html. – Заголовок з екрана.

2. Зуев Е. А. Языки программирования: критерии выбора. [Електронний ресурс] / Е. А. Зуев // IT ШКОЛА SAMSUNG — Режим доступу: https://www.youtube.com/watch?v=T70qJndjYi0. – Заголовок з екрана.

3. Производительность C++ vs. Javavs. PHP vs. Python. Тест «в лоб» [Електронний ресурс] – Режим доступу: https://habrahabr.ru/post/66562/. – Заголовок з екрана.

УДК 539.3

# НАПРУЖЕНИЙ СТАН ОРТОТРОПНОЇ ОБОЛОНКИ З ОТВОРОМ І ТРІЩИНАМИ

***В.А. Врублевський***

В наш час конструкції, які містять в собі тонкостінні оболонки, мають широке застосування у гідротехніці, літако- та ракетобудуванні, машинобудуванні, будівництві та інших галузях. Одним із головних напрямків розвитку цих галузей є збільшення міцності конструкцій та зниження їх ваги.

Однак такий крок призводить до різкого зниження ефекту пластичності цих оболонок і здатності до крихкого руйнування. Їх надійність нерідко залежить від наявності різного роду дефектів: тріщин, отворів, зношення та ін. Тому дослідження напруження ортотропних оболонок з отворами і тріщинами має досить великий теоретичний та практичний зміст.

Подібні дослідження проводились для ізотропних оболонок, а також частково для анізотропних оболонок з отворами. В той час, як ортотропні оболонки з отвором і тріщинами вивчені недостатньо.

**Мета.** Метою дослідження є розвиток методики розв'язування задач про напружений стан ортотропних оболонок, які містять отвір та тріщини та визначення основних закономірностей впливу геометричних і механічних параметрів на збурений напружений стан оболонок.

**Завдання.** Для досягнення поставленої мети передбачено вирішення наступних завдань:

– побудувати систему граничних інтегральних рівнянь, що описують напружений стан ортотропної оболонки з отвором та однією тріщиною;

– побудувати систему граничних інтегральних рівнянь, що описують напружений стан ортотропної оболонки з отвором та двома паралельними тріщинами;

– побудувати систему граничних інтегральних рівнянь, що описують напружений стан ортотропної оболонки з отвором та двома довільними тріщинами.

**Методи дослідження.** Сформульовані завдання мають бути розв'язані методом сингулярних інтегральних рівнянь. В свою чергу, розв'язки сингулярних інтегральних рівнянь мають бути отримані за допомогою методу механічних квадратур з використанням невідомих функцій.