УДК 004.438

Грисюк Андрій, ст. IV курсу факультету кібернетики; науковий керівник – к.пед.н., доцент Лотюк Ю. Г. (Міжнародний економіко-гуманітарний університет імені академіка Степана Дем'янчука, м. Рівне)

НОВІ МОЖЛИВОСТІ ВИКОРИСТАННЯ МОВ ЕЗОТЕРИЧНОГО ПРОГРАМУВАННЯ ТИПУ MALBOLGE

Анотація. *У статті досліджено езотеричну мову програмування Malbolge. Розглянуто специфіку компіляції програм цією мовою та алгоритми роботи. Розкрито зв'язок мови Malbolge з машиною Тюрінга, використання трійкової арифметики та криптографічних засобів мови. Обґрунтовано доцільність використання відомостей з галузей вищої математики та математичної логіки, програмування та криптології, які дозволяють створити оригінальну програму на Malbolge. Показано, що створення програм цією мовою розвиває у студентів алгоритмічне мислення, поглиблює знання з програмування та математичної логіки, значно полегшує вивчення криптологічних алгоритмів та їх використання в навчальному процесі.*
Ключові слова: *програмування, Malbolge, езотеричне програмування, алгоритм роботи Malbolge, трійкова арифметика.*

Аннотация. *В статье исследован эзотерический язык программирования Malbolge. Рассмотрены специфика компиляции программ на этом языке и алгоритмы работы. Рассмотрена связь языка Malbolge с машиной Тьюринга, использование троичной арифметики и криптографических средств языка. Обосновано целесообразность использования сведений из области высшей математики и математической логики, программирования и криптологии, позволяющих создать оригинальную программу на Malbolge. Показано, что создание программ на этом языке развивает у студентов алгоритмическое мышление, углубляет знания по программированию и математической логике, значительно облегчает изучение криптологических алгоритмов и их использование в учебном процессе.*
Ключевые слова: *программирование, Malbolge, эзотерическое программирование, алгоритм работы Malbolge, троичная арифметика.*

Annotation. *The article investigates esoteric programming languages Malbolge. The specificity of compiling programs on that language and its algorithms are considered. Particular attention is paid to the links of Malbolge language with the Turing machine, using ternary arithmetic, cryptographic language. In the research, the details of the areas of Mathematics and Mathematical Logic, programming and cryptology that allow setting up Malbolge original*

application are demonstrated. It is shown that the creation of programs in that language develops students' algorithmic thinking, deepens knowledge of programming and mathematical logic, considerably facilitates study of cryptology algorithms and their use in the classroom.
Keywords: *programming, Malbolge, esoteric programming, Malbolge algorithm, ternary arithmetic.*

Езотеричні мови програмування мають великі перспективи розвитку, оскільки дозволяють швидко та якісно програмувати криптографічні алгоритми. Наукова новизна нашої розробки полягає у тому, що такі мови програмування вперше застосовані для вирішення задач криптографічного захисту даних.

Актуальність нашого дослідження полягає в можливостях широкого використання сучасних криптографічних алгоритмів у навчальному процесі та демонстрації їх можливостей для студентів.

Історія езотеричних мов програмування починається з робіт Бена Олмстеда, розпочатих у 1998 році [1; 2]. Мова розроблена так, щоб бути максимально важкою для написання програм і отримала свою назву від восьмого кола пекла, описаного в Данте [1]. Першою програмою, яка була бути написана новою мовою програмування, є програма «Hello, world». На цей час відомо 3 варіанти [3] зазначеної програми (рис.1), перша з яких була написана під назвою Lisp через два роки після появи мови.

1)
```
(=<`:9876Z4321UT.-
Q+*)M'&%$H"!~}|Bzy?=|{z]KwZY44Eq0/{mlk**hKs_dG5[m_B
A{?-Y;;Vb'rR5431M}/.zHGwEDCBA@98\6543W10/.R,+O<
```

2)
```
('&%:9]!~}|z2Vxwv-
,POqponl$Hjig%eB@@>}=<M:9wv6WsU2T|nm-
,jcL(I&%$#"`CB]V?Tx<uVtT`Rpo3NlF.Jh++FdbCBA@?]!~|4Xz
yTT43Qsqq(Lnmkj"Fhg${z@>
```

3)
```
(=<`#9]~6ZY32Vx/4Rs+0No-
&Jk)"Fh}|Bcy?`=*z]Kw%oG4UUS0/@-ejc(:'8dc
```

Рис. 1. Три варіанти програми «Hello, world»

Всі ці програми виконують одну і ту ж дію, виводять на екран «Hello, world», єдиною різницею при застосуванні цих програм є алгоритм виконання, від якого залежить кількість символів у коді програми.

Однак, поки що не досліджувалося питання, який саме алгоритм раціональніший [4; 5], так як на цей час існує 7 відомих нам програм

написаних на Malbolge [2]. Відомо, що існує як мінімум 8 алгоритмів шифрування, які описав Лу Шеффер у 2002 році [3]. 17 серпня 2004 року Томаш Вегжановски написав генератор програм, який виводив задані рядки [3], однак, програми отримані таким способом, були громіздкими.

Метою нашої статті є дослідження досвіду створення власних прикладних програм мовою Malbolge, аналіз їх виконання та застосування у навчальному процесі.

Malbolge – **це машинна** мова, яка працює в трійковій системі числення, тобто має 3 регістри: a, c, d. Регістр a – є акумулятором, який використовується для маніпулювання даними, регістр c – використовується як вказівник на поточну команду, регістр d – використовується для управління даними [2; 3]. Існує лише *8 команд:*

1) *j* – проводить операцію jmp [d+1];
2) *i* – встановлює код покажчика на значення в комірці та набуває поточних даних;
3) *p* – проводить операцію crz [d,a];
4) ***** – проводить операцію rotr [d];
5) < – виводить дані на екран;
6) / – читає значення, перетворює його в трінарне та зберігає в регістрі a;
7) *v* – вказує на закінчення виконання програми;
8) *o* – проводить операцію NOP;

та *4 операції:*

1) rotr [d] – виконує трітну операцію за формулою rotr [d] = (a/3+a%3* 19683);
2) NOP – операція переходу на наступну комірку, збільшує c та d на 1;
3) jmp[d+1] – переходить на комірку, на яку вказує d+1;
4) crz – одна з найкорисніших операції, яка виконується над регістром a та d: crz [d,a], виконується згідно таблиці (рис. 2):

	00	01	02	10	11	12	20	21	22
00	04	03	03	01	00	00	01	00	00
01	04	03	05	01	00	02	01	00	02
02	05	05	04	02	02	01	02	02	01
10	04	03	03	01	00	00	07	06	06
11	04	03	05	01	00	02	07	06	08
12	05	05	04	02	02	01	08	08	07
20	07	06	06	07	06	06	04	03	03
21	07	06	08	07	06	08	04	03	05
22	08	08	07	08	08	07	05	05	04

Рис. 2. Таблиця істинності операції *crz*

Коли програма завантажується в пам'ять комп'ютера, то вона перевіряється на присутність команд (шифрується), і якщо хоча б один символ не відповідає команді програма аварійно завершується.

Детальніше процес виконання програми розглянемо на прикладі написаної нами програми (рис. 3):

```
(=<;:^!7[54321654-Q+*N;-,+*)5'&%$#"!-}|{z]rwIoGVlDTBhPlk*<LKf_
```

Рис. 3. Авторська програма «Hello, world»

При завантаженні програми у пам'ять комп'ютера вона перевіряється на відповідність команд синтаксису мови – у вхідному коді не повинно бути нічого крім команд. Процес перевірки відбувається таким чином [2]:

1) вхідний код ділиться на символи;
2) здійснюється шифрування за формулою: (m[c]+c-33)%94, остача від ділення за цією формулою береться як індекс символу в наступному рядку (індексація починається з нуля);
3) здійснюється дія відповідно по команди;
4) виконується наступна команда у програмі;
5) якщо поточний символ не вказує на завершення програми, то переходимо до пункту 1, інакше закінчуємо виконання програми.

Наведемо приклад покрокового виконання такої програми (рис. 3):

Першим символом є (=40 (код ASCII). Далі переходимо до пункту 2 алгоритму: c = 0, отже, (40+0-33)%94 = 0,07. Використовуємо остачу від попереднього ділення, як індекс символу в рядку. А самим символом є: j – команда jmp [d+1].

Переходимо до пункту 3 алгоритму: d = (=40, тобто, jmp[d+1] = 41.

Потім переходимо на 41 елемент нашого вхідного коду, яким є] =93 (код ASCII). Для регістру d присвоюємо результат команди: d =]=93.

Переходимо до 4 пункту алгоритму: Перевіряємо чи команда не є завершенням роботи програми.

Командою завершення є *v*, в нашому випадку командою була *j*, тобто, умова виконується і виконання переходить до 1 пункту алгоритму.

Виконуємо другий крок: вхідним символом є =, що відповідає 61 коду таблиці ASCII; шифруємо символ, c = 1: (61+1-33)%94=0,30, 30 символом є *p*, який відповідає команді crz; виконуємо поточну команду, на даному етапі виконання: a = 0, a d = 93, де a = crz(a,d) = (0,93) = T=84 (код ASCII).

Команда, яку ми виконували, не є завершальною, тому переходимо до наступного символу вхідного рядка. Продовжуємо виконання, поки поточний символ не вкаже на завершення програми.

Нашу програму (рис. 3), можна виконати без шифрування: переводимо всі символи в команди та виконуємо команди.

Коротко опишемо цей алгоритм: переведемо весь наш вхідний рядок в команди згідно формули (m[c]+c-33)%94 та рядка шифрування; виконуємо команди доки не зустрінемо символ завершення *v,* причому, до уваги беруться не графічні символи, а їх код; при виконанні поточної команди p = crz(a,d), результат записується у регістр a, при цьому змінюється символ **у** вхідному рядку на результат a = 0; a = crz(a,d) = crz(0,93) = 84 =T.

Збільшуємо *c* та *d* на 1; виконуємо наступну команду. Результатом якої є p = crz(a,d). Цей результат записується у регістр *a,* а також змінюється символ у вхідному рядку на результат a = 93; a = crz(a,d) = crz(93,114) = \x06.

Коли результат операції crz менший за 32 (найменший графічний символ в таблиці ASCII), то записується такий результат: \x06 – де 06 тлумачиться, як 6, а решта символів ігнорується.

При такому результаті в регістр *a* записується цифровий результат, а у вхідний рядок записується символ: «.» . В результаті виконання цих команд отримаємо: a = \x06. Збільшуємо регістри *c* та *d* на 1 та виконуємо наступну команду. Отримаємо a = \x06, a = crz(a,d) = crz(\x06,119) = 82 =R.

Збільшуємо регістри *c* та *d* на 1, виконуємо наступну команду; отримуємо a = 82; a = crz(a,d) = crz(82,73) = 77 =M.

Збільшуємо регістри *c* та *d* на 1, виконуємо наступну команду, яка здійснює вивід значення регістра *a* на екран. Результатом виконання цієї програми є символи: «MegU».

Мова Malbolge не повністю відповідає тьюринг архітектурі через обмеження по оперативній пам'яті. Тому було здійснено кілька спроб створити тьюринг у повні версії Malbolge (Malbolge-T) [6]. Це були теоретичні варіанти Malbolge, що перенаправляють потік вводу-виводу. Таке перенаправлення дозволяє зняти з програм обмеження по оперативній пам'яті. Malbolge-T сумісний з Malbolge. Тьюринг-повний варіант компілятора дозволяє виконувати програми будь-якої довжини.

За результатами проведеного дослідження відомостей з галузей вищої математики та математичної логіки, програмування та криптології було створено оригінальну програму мовою Malbolge. Створення програм цією мовою розвиває алгоритмічне мислення, поглиблює знання студентів з програмування та математичної логіки. Можливе успішне прикладне використання програм мовою Malbolge при вивченні криптологічних алгоритмів. Програмування мовою Malbolge значно полегшує засвоєння студентами трійкової арифметики.

1. Frederic P. Miller. Malbolge Esoteric programming language, Public domain, Inferno (Dante) / P. Miller Frederic, F. Vandome Agnes, McBrewster John // Alphascript Publishing. – 108 p. **2.** Jesse Russell. Malbolge / Russell Jesse // Alphascript Publishing – 56 p. **3.** Jesse Russell. Malbolge / Russell Jesse, Cohn Ronald. – 2013. – 104 p. **4.** Уэзерелл Ч. Этюды для программистов / Ч. Уэзерелл, М. : «Мир», 1982. Уэзерелл Ч., 59 с. **5.** Хофштадтер Дуглас. «Гедель, Эшер, Бах: эта бесконечная гирлянда» / Дуглас Хофштадтер. – Издательство : Бахрах-М. – 2001.