УДК: 004.438

Грисюк Андрій, ст. магістратури факультету кібернетики; науковий керівник – д ф.-м.н., професор Джунь Й. В. (Міжнародний економіко-гуманітарний університет імені академіка Степана Дем'янчука, м. Рівне)

ОСОБЛИВОСТІ ЗАСТОСУВАННЯ НАЙБІЛЬШ СКЛАДНОЇ МОВИ ПРОГРАМУВАННЯ MALBOLGE

Анотація. *У статті досліджено одну з езотеричних та найскладніших мов програмування, Malbolge. Розглянуто специфіку компіляції програм цією мовою, алгоритми роботи. Вивчено зв'язок мови Malbolge з машиною Тюрінга, використання трійкової арифметики, криптографічні засоби мови. Використано відомості з галузей вищої математики та математичної логіки, програмування та криптології, які дозволяють створити оригінальну програму на Malbolge. Обгрунтовано, що створення програм цією мовою розвиває у студентів алгоритмічне мислення, поглиблює знання з програмування та математичної логіки.*
Ключові слова: *програмування, езотеричне програмування, Malbolge, алгоритм роботи Malbolge, трійкова арифметика.*

Аннотация. *В статье исследована одну из езотеричных и сложных языков программирования Malbolge. Рассмотрена специфика компиляции программ на этом языке, алгоритмы работы. Изучена связь языка Malbolge с машиной Тьюринга, использование троичной арифметики, криптографические средства языка. Использованы данные из разных отраслей высшей математики и математической логики, программирования и криптологии, которые позволяют создать приложения на Malbolge. Обосновано, что создание программ на этом языке развивает у студентов алгоритмическое мышление, углубляет знания с программирования и математической логики.*
Ключевые слова: *программирование, эзотерическое программирования, Malbolge, алгоритм работы Malbolge, троичная арифметика.*

Annotation. *The article investigates one of the most complex and esoteric programming languages, Malbolge. The specificity compiling programs that language algorithms. Special attention is paid to the connection of language Malbolge 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 you to set up the original application Malbolge. Create applications that language develops in students of cybernetics algorithmic thinking and deepens knowledge of programming and mathematical logic. The program facilitates language Malbolge study Cryptologic algorithms and their use in the classroom.*

Keywords: *programming, esoteric programming, Malbolge, algorithm Malbolge, ternary arithmetic.*

Мова програмування Malbolge має широкі перспективи розвитку, оскільки дозволяє швидко та якісно програмувати криптографічні алгоритми. Наукова новизна розробленого алгоритму полягає у тому, що він дав можливість її застосування для вирішення конкретних задач криптографічного захисту даних.

Актуальність теми полягає в можливості широкого використання сучасних криптографічних алгоритмів у навчальному процесі, демонстрації їх можливостей для студентів магістратури факультету кібернетики.

Історія мови Malbolge починається з робіт Б. Олмстеда [1; 2]. Специфіка мови полягає в тому, що вона є максимально важкою для написання програм [1]. За традицією, початковою програмою, має бути «Hello, World». Зараз відомо 3 варіанти цієї програми [1] (перша з яких була написана під назвою Lisp).

Зазначені програми виконують одну і ту ж дію, виводять на екран «Hello, World». Різницею в застосуванні цих програм є алгоритм виконання.

В наш час не відомо, який саме алгоритм раціональніший [3], оскільки існує 7 відомих програм написаних на Malbolge [2]. Відомо, що існує як мінімум 8 алгоритмів шифрування, які описав Лу Шеффер у 2002 році [3]. У 2004 році Т. Вегжановски [3] написав генератор програм, який виводив задані рядки, однак, програми отримані таким способом, були громіздкими.

Метою нашої статті є створення власної програми мовою Malbolge.

Malbolge – це мова, яка працює в трійковій системі числення, та містить 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]. Виділимо 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 данні 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.

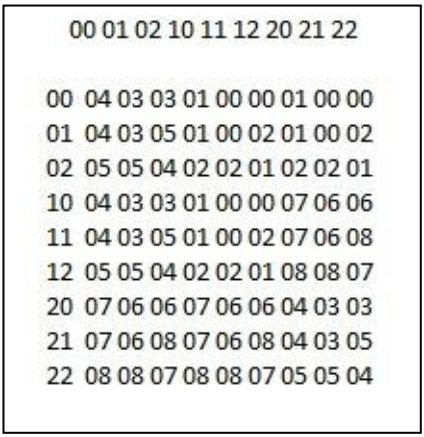
та 4 операції:

1) rotr[d] – виконує тернарну операцію за формулою rotr[d] = (a/3+a%3* 19683);

2) NOP – операція переходу в наступну комірку, збільшує c та d на 1;

3) jmp[d+1] – переходить на комірку, на яку вказує d+1;

4) crz – виконується над регістром a та d, згідно таблиці (рис. 1):



	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. 1. Таблиця істинності операції **crz.**

При завантаженні програми, перевіряється присутність команд (шифрується), і якщо хоча б один символ не відповідає команді, програма завершується.

Процес виконання розглянемо на прикладі написаної нами програми (рис. 2):

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

Рис. 2. Програма «Megu forever»

При завантаженні програми перевіряється на відповідність команд до 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, залишок від ділення за цією формулою береться як індекс символу в наступному рядку (індексація починається з нуля);

3) Здійснюється дія відповідно до команди;

4) Виконується наступна команда у програмі;

5) Якщо поточний символ не вказує на завершення програми, то здійснюється перехід до пункту 1, інакше закінчуємо виконання програми.

Покрокове виконання авторської програми (рис. 2):

Першим символом є (=40 (код ASCII). Далі слід перейти до пункту 2 алгоритму: c = 0, отже, (40+0-33)%94 = 0,07. Дальше використовується залищок від попереднього ділення, як індекс символу в рядку. А самим символом є: j - команда jmp[d+1]. Пункту 3 алгоритму: d = (=40, тобто, jmp[d+1] = 41.

Потім необхідно перейти на 41 елемент вхідного коду, яким є] =93 (код ASCII). Для регістру d присвоюється результат команди: d =]=93.

Пісоля цього здійснюється перехід до 4 пункту алгоритму: перевіряється чи команда не є завершенням роботи програми.

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

Виконується другий крок: вхідним символом є =, що відповідає 61 коду таблиці ASCII. Шифрується символ c = 1: (61+1-33)%94=0,30. 30 символом є *p,* який відповідає команді crz. Виконується поточна команда на цьому етапі виконання: a = 0, a d = 93, де a = crz(a,d) = (0,93) = T=84 (код ASCII).

Виконана команда не є завершальною, тому потрібно перейти до наступного символу вхідного рядка. При цьому продовжується виконання, поки поточний символ не вкаже на завершення програми.

Програму, зображену на рис. 2, можна виконати без шифрування Для цього необхідно перевести всі символи в команди та виконати команди.

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

Збільшивши *c* та *d* на 1; виконаємо 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.

Якщо результат операції crz менший за 32 (найменший графічний символ в таблиці ASCII), то записується результат: \x06, де 06 тлумачиться, як 6.

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

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

Збільшивши регістри *c* та *d* на 1, виконаємо наступну команду, яка здійснює вивід значення регістра *a* на екран. Результатом виконання цієї програми є символи: «Megu *forever*».

Malbolge не повністю відповідає машині Тьюринга. Тому було здійснено кілька спроб створити версії Malbolge, які будуть відповідати повноті по

Тьюрінгу [6]. Це були теоретичні варіанти Malbolge, що перенаправляють потік вводу-виводу. Таке перенаправлення дозволяє зняти з програм обмеження по оперативній пам'яті.

В нашій роботі на основі відомостей з вищої математики та математичної логіки, програмування та криптології, створена *оригінальна* програма мовою Malbolge. Створення програм цією мовою істотно розвиває алгоритмічне мислення, поглиблює знання студентів з програмування та математичної логіки. Можливе успішне прикладне використання програм мовою Malbolge при вивченні криптологічних алгоритмів.

1. Malbolge [Електонний ресурс] // Режим доступу : ru.wikipedia.org/wiki. **2.** Miller F. P. Malbolge. Esoteric programming language / F. P. Miller, A. F. Vandome. – London : Public domain, 2014. – 108 p. **3.** Russell J. Malbolge. Alphascript Publishing / J. Russell. – Oxford : ATB, 2013. – 185 p. **4.** Уэзерелл Ч. Этюды для программистов / Ч. Уэзерелл. – М. : Мир, 1982. – 329 с. **5.** Дуглас Х. Эта бесконечная гирлянда / Х. Дуглас. – М. : Бахрах-М, 2001. – 588 с. **6.** Грисюк А. В. Нові можливості використання мов езоторичного програмування типу Malbolge / А. В. Грисюк. – Збірник студентських наукових праць. – № 2. – Рівне : РВУ МЕГУ ім. акад. С. Дем'янчука, 2014. – С. 261–266.