МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки комп’ютерної та програмної інженерії

Кафедра безпеки інформаційних технологій

**Blockchain-технології**

**Лабораторна робота №5**

Тема: Алгоритм Луна

Тривалість заняття: 180 хв.

Київ 2023

**Тема:** Алгоритм Луна.

**Мета:** Ознайомитись та навчитись працювати з алгоритмом Луна.

**Теоретичні відомості**

Алгоритм Луна (англ. Luhn algorithm), або формула Луна (англ. Luhn formula), також відомий під назвою «modulus 10» або «mod 10», це проста формула перевірки контрольної суми, що використовується для валідації різноманітних ідентифікаційних номерів, таких як номери кредитних/платіжних карток, номери IMEI, американських National Provider Identifier Number, канадських Canadian Social Insurance Number, ізраїльських ID Numbers та грецьких Social Security Numbers (ΑΜΚΑ). Алгоритм був створений науковцем з IBM Гансом Петером Луном[en] і описаний патентом U.S. Patent No. 2,950,048, описаним 6 січня 1954 року та схваленим 23 серпня 1960 року.

Алгоритм є публічним і широко використовується. Він також вказаний у ISO/IEC 7812[en]. Цей алгоритм не створювався як криптографічно надійна хеш-функція, а суто як захист від випадкових помилок. Більшість кредитних карток і багато урядових ідентифікаційних номерів використовують цей алгоритм як простий метод відсіювання неправильних номерів.

У світі електронних платежів та онлайн-транзакцій, де безпека є першорядним завданням, алгоритми перевірки валідності номерів кредитних карток відіграють важливу роль. Один із найбільш широко використовуваних алгоритмів, який допомагає не тільки запобігти помилкам при введенні номерів карт, але й виявити можливі спроби шахрайства, — це алгоритм Місяць, також відомий як «алгоритм Luhn».

Алгоритм Луна був розроблений математиком Хансом Петером Луном у 1954 році і з того часу став стандартним методом перевірки валідності номерів кредитних карток у багатьох платіжних системах. Він базується на простій математичної операції та контрольній сумі, яка обчислюється з цифр номера картки.

Формула перевіряє номер за його контрольною цифрою, котра зазвичай додається до часткового номера акаунту (partial account number) при генеруванні повного номера акаунту (full account number). Цей номер повинен успішно проходити таку перевірку:

* Починаючи від крайньої правої цифри (контрольної), рухаємося ліворуч, подвоюючи кожну другу цифру. Якщо результат подвоєння є більшим, ніж 9 (як-от: 8 × 2 = 16), то потрібно додати числа результату (наприклад: 16: 1 + 6 = 7, 18: 1 + 8 = 9), або, як альтернатива — відняти 9 від результату (16: 16 − 9 = 7, 18: 18 − 9 = 9).
* Знайти суму всіх цифр.
* Якщо ділення суми на 10 не має остачі (сума закінчується нулем,) то номер є правильним відповідно до формули Луна.

Нехай у нас є номер «7992739871», який матиме контрольну цифру в наступному вигляді: 7992739871x. Тоді:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Номер** | 7 | 9 | 9 | 2 | 7 | 3 | 9 | 8 | 7 | 1 | **x** |  |
| **Подвоїти через одну** | 7 | **18** | 9 | **4** | 7 | **6** | 9 | **16** | 7 | **2** | **x** |
| **Додати цифри** | 7 | **9** | 9 | 4 | 7 | 6 | 9 | **7** | 7 | 2 | **x** |  |

Сума всіх цифр у третьому рядку дорівнюватиме **67+x**.

Контрольну цифру (x) можна отримати множенням суми цифр на 9 і діленням результату на 10 ((67 × 9) mod 10). У формі алгоритму:

* Обчислити суму цифр без контрольної (67).
* Помножити на 9 (603).
* Остання цифра (3) буде контрольною цифрою, як-от **x=3**.

Альтернативний метод: контрольна цифра (x) отримується в результаті суми решти цифр (3 рядок), ділення її на 10 з остачею, і віднімання цієї остачі від 10 (остача (67/10) = 7; 10 − 7 = 3). В алгоритмічній формі:

* Вирахувати суму цифр без контрольної (67).
* Отримати остачу (7).
* Відняти остачу від 10.
* Результат (3) є контрольною цифрою. Якщо сума закінчується нулем, то нуль буде контрольною цифрою.

Відповідно повний номер акаунту матиме вигляд 79927398713.

Кожен з номерів 79927398710, 79927398711, 79927398712, 79927398713, 79927398714, 79927398715, 79927398716, 79927398717, 79927398718, 79927398719 може пройти перевірку наступним чином:

* Подвоїти цифри через одну, починаючи з крайньої правої: (1×2) = 2, (8×2) = 16, (3×2) = 6, (2×2) = 4, (9×2) = 18
* Додати всі *окремі* цифри (цифри в дужках є результатами кроку 1): x (контрольна цифра) + (2) + 7 + (1+6) + 9 + (6) + 7 + (4) + 9 + (1+8) + 7 = x + 67.
* Якщо сума ділиться без остачі на 10, то номер може бути правильним. Зверніть увагу, що 3 — це єдина правильна контрольна цифра, котра поверне суму (70), яка ділиться на 10 без остачі.
* Отже всі ці номери є неправильними, окрім 79927398713, котрий має правильну контрольну цифру.

Ви можете також використати той самий алгоритм створення контрольної суми, припускаючи що наявна перевірка контрольної суми насправді відсутня. Тоді порахуйте контрольну суму і порівняйте результат із оригінальною контрольною сумою, наявною в кредитної картки. Якщо оригінальна контрольна сума відповідає порахованій, то номер є правильним.

Алгоритм Луна може знайти помилку в одній цифрі, так само як і всі перестановки сусідніх цифр, проте він не може виявити перестановку двоцифрових послідовностей 09 - 90 (чи навпаки). Він виявлятиме також 7 із 10 можливих помилок дублювання (але не виявить 22 ↔ 55, 33 ↔ 66 або 44 ↔ 77).

Інші, складніші алгоритми перевірки контрольної цифри (такі як алгоритм Вергуффа чи алгоритм Дамма) можуть знаходити більше помилок. Існує ще алгоритм Луна mod N (Luhn mod N algorithm), який є розширенням звичайного і додає підтримку нецифрових рядків.

Оскільки алгоритм оперує цифрами справа наліво і нулі впливають на результат тільки тоді, коли вони викликають зміщення позиції, то нулі на початку послідовності цифр не вплинуть на розрахунок. Внаслідок цього системи, які мають на початку певну кількість цифр (наприклад при конвертуванні 1234 в 0001234), можуть проходити перевірку алгоритмом Луна з нулями або без них і видавати однакові результати.

Додавання 0 до номерів непарної довжини дозволяє перевіряти номери зліва направо, а не навпаки, з подвоєнням цифр на непарних місцях.

Алгоритм з'явився в американському патенті для ручного механічного пристрою для обрахунку контрольних сум, а тому основною вимогою до алгоритму була відносна простота реалізації. Механізм пристрою ділив суму на 10, але цифри від подвоєння та наступної редукції не отримувалися механічним шляхом, а позначалися у зміненому порядку на самому пристрої.

При внесенні даних в інформаційні системи суспільство досить часто робить помилки. Особливо це стосується числових даних, таких як: номер кредитної карти чи ІМЕІ (міжнародний ідентифікатор мобільного обладнання) мобільного телефону.

Алгоритм Луна допомагає перевірити, чи було зроблено помилку при веденні номеру, який містить контрольну цифру Луна. При введенні номеру банківської карти для оплати в мережі Інтернет, платіжна система спочатку перевіряє контрольну суму, а потім, якщо вона відповідає дійсності, система відправляє дані на опрацювання в банк.

Для того, щоб підрахувати контрольну цифру, використовують так звану формулу Луна. Вона використовується в:

номерах банківських карт;

IMEI;

ICCID (унікальний серійний номер сім карти).

Дана формула перевіряє номер за його контрольною сумою або цифрою. Це залежить від особливості того номеру, який перевіряється (для кредитних карток – це сума, для номерів ІМЕІ – це цифра).

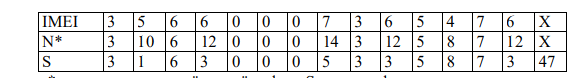
Наведемо приклад розрахунку контрольної цифри для номеру IMEI певного телефону. Наприклад, IMEI: 356600073654763 – остання цифра (це і є контрольна цифра Луна) – 3. Для того, щоб обрахувати контрольну цифру Луна для номера IMEI потрібно:

1. Провести подвоєння кожної другої цифри даного номера.

2. Додати всі цифри.

3. Обрахувати цифру Луна за формулою: X=(S\*9) mod 10. У таблиці 1 наведено перевірку контрольної цифри Луна.

Таблиця 1 – Перевірка контрольної цифри Луна



\*- подвоєння кожної другої цифри, S- сума цифр.

Підставивши значення S, отримаємо:

X=(47\*9) MOD 10 = 3

Наведемо приклад перевірки контрольної суми номеру для кредитної карти. Для того, щоб обрахувати контрольну цифру Луна для номера кредитної картки потрібно:

1. Провести подвоєння кожної другої цифри даного номера.

2. Додати всі цифри.

3. Перевіряємо, чи отримана сума ділиться націло на 10.

У таблиці 2 наведено приклад такої перевірки.

Таблиця 2 – Перевірка контрольної суми номеру кредитної карти



\* - подвоєння кожної другої цифри.

В даному випадку сума всіх цифри дорівнює 70, а це число ділиться націло на 10, отже, номер картки вірний.

Використання даного алгоритму не виключає декількох недоліків: він шукає помилки в одній цифрі на належному рівні, а якщо є помилки в декількох цифрах, то контрольна цифра може співпасти з великою ймовірністю. Але цей алгоритм, на відміну від інших алгоритмів пошуку контрольних цифр, потребує мінімум ресурсів для підрахунку.

Приклад реалізації алгоритму Луна на мові програмування С :

#include <stdio.h>

**enum** CardType

{

  type\_master\_card,

  type\_visa,

  type\_american\_express,

  type\_other

};

static char\* EmitentNames[] = {"Master Card", "VISA", "American Express", "Other"};

int GetFirstTwoDigits(const int\* const cardNumber)

{

  return (cardNumber[0] \* 10 + cardNumber[1]);

}

**enum** CardType GetCardType(const int\* const cardNumber)

{

**enum** CardType type;

  int firstTwoDigits = GetFirstTwoDigits(cardNumber);

  if ((firstTwoDigits >= 51) && (firstTwoDigits <= 55))       type = type\_master\_card;

  else if (firstTwoDigits / 10 == 4)                          type = type\_visa;

  else if ((firstTwoDigits == 34) || (firstTwoDigits == 37))  type = type\_american\_express;

  else                                                        type = type\_other;

  return type;

}

int Luhn(const int\* const cardNumber, const int cardNumberLength)

{

  int validationFlag = 0;

  int p = 0;

  int sum = 0;

  int i = 0;

  for (i = cardNumberLength - 1; i > -1; i--)

  {

    p = cardNumber[i];

    if ((i + 1) % 2 == 0) p = p \* 2;

    if (p > 9) p = p - 9;

    sum = sum + p;

  }

  if (sum % 10 == 0) validationFlag = 1;

  return validationFlag;

}

void VerifyCard(const int\* const cardNumber, const int cardNumberLength)

{

  if (Luhn(cardNumber, cardNumberLength))

    printf("This is a VALID credit card. Emitent: %s**\n**", EmitentNames[GetCardType(cardNumber)]);

  else

    printf("This card is INVALID!**\n**");

}

int main(void)

{

  int card[] = {3, 7, 8, 7, 3, 4, 4, 9, 3, 6, 7, 1, 0, 0, 0};

  VerifyCard(card, sizeof(card)/ sizeof(\*card));

  return 0;

}

**Завдання**

1. Розрахуйте по прикладу (з використанням алгоритму Луна) в теоретичних відомостях :
2. Контрольну цифру Луна для IMEI ( IMEI 154789445003253)
3. Контрольну суму номеру кредитної картки ( картка 1231456745895503)
4. Спосіб розв’язання та отримані результати занесіть до звіту
5. Реалізуйте алгоритм Луна на одній з мов програмування, приклад реалізації є в теоретичних відомостях. Код розробленої вами програми повинен відрізнятись від коду який є в прикладі.
6. Реалізацію програмно алгоритму Луна занести до звіту разом зі скріншотами роботи розробленої вами програми та програмним кодом.
7. Дайте відповідь на контрольні питання.
8. Зробіть висновки та занесіть їх до звіту.

**Контрольні питання**

1. Що таке алгоритм Луна?
2. Для чого використовують алгоритм Луна?
3. Ким був розроблений алгоритм Луна?
4. В якому році алгоритм Луна був розроблений та схвалений?
5. Яким чином працює алгоритм Луна?