Функціональні і логічні мови називають декларативними, або непроцедурного, оскільки програма являє собою не набір команд, а опис дій, які необхідно здійснити. Цей підхід істотно простіше і прозоріше формалізується математичними засобами. Отже, програми простіше перевіряти на наявність помилок (тестувати), а також на відповідність заданій технічної специфікації (верифікувати). Високий ступінь абстракції також є перевагою даного підходу. Фактично програміст оперує не набором інструкцій, а абстрактними поняттями, які можуть бути досить узагальненими.

Список використаних джерел

1. Угринович Н.Д. Информатика и информационные технологии. Учебник 10-11 класс. М.: Лаборатория Базовых Знаний, 2001.

2.https://uk.wikipedia.org/wiki/%D0%86%D1%81%D1%82%D0%BE%D1%80%D1%96%D1%8F_%D0%BC%D0%BE%D0%B2_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F

3.https://stud.com.ua/97344/informatika/klasifikatsiya_programuvannya

4.https://www.typesnuses.com/types-of-programming-languages-with-differences/

5.https://dou.ua/lenta/articles/language-rating-jan-2019/

УДК 621.923.42

ОСНОВНІ СХЕМИ АЛГОРИТМІВ СТИСНЕННЯ ДАНИХ

Сазонов В. С., студ. гр. ВТ-161
Науковий керівник: **Степенко С. А.**, к.т.н.
Національний університет «Чернігівська політехніка»

Метод стиснення JPEG-LS заснований на алгоритмі стиснення LOCO-I. Він дуже відрізняється від JPEG та JPEG2000, хоча схожий. Цей алгоритм рекомендується використовувати для систем із обмеженим вмістом ресурсів, наприклад космічних станцій на камерах Chibe. Для прогнозування значення звичайного пікселя x використовуються пікселі контексту a, b, c, d. Залежно від контексту, він вибирає режим: послідовний (режим запуску) або звичайний (звичайний режим). Послідовний режим вибирається, якщо y і z швидше збігаються, регулярний - якщо. У разі використання послідовного режиму кодер робота починає переглядати поточний порядок пікселів x і знаходить найбільшу довжину серії пікселів, що збігаються з контекстним пікселем a. Тому в межах поточного рядка отримують послідовність однакових пікселів, які збігаються з позначенням, відомим пікселем a. Після цього кодується довжина послідовності. При використанні звичайного режиму значення пікселів a, b і c використовуються для обчислення прогнозованого значення пікселя x (Px). Також називається помилка прогнозу Errval. Його значення дорівнює різницевим значенням x і Px. Помилка Errval виправляється, а потім кодується за допомогою кодів Pigeon. Код Голомба залежить від a, b, c, d та Errval цих пікселів.

Код Хаффмана (Huffman code) це мінімальнонадлишковий префіксний код (minimum-redundancy prefix code). Розглянемо основні ідеї коду Хаффмана та зробимо дослідження ряду важливих властивостей алгоритму. Основною ідеєю алгоритму Хаффмана є те, що кодування символів вхідного алфавіту здійснюється різним числом біт. Символи, які зустрічаються частіше, будуть закодовані меншим числом біт, ніж ті, які зустрічаються рідше. Отриманий код буде оптимальний або, іншими словами, мінімально-надлишковий. Ідея алгоритму була опублікована Девідом Хаффманом в 1952 році. Алгоритм Хаффмана двопрохідний. На першому проході будується частотний словник і генеруються коди. На другому проході відбувається безпосередньо кодування. За 50 років з дня опублікування, код Хаффмана нітрохи не втратив своєї актуальності і значущості. Так з упевненістю можна сказати, що ми стикаємося з ним, в тій чи іншій формі (справа в тому, що код Хаффмана рідко використовується окремо, частіше працюючи у зв'язці з іншими алгоритмами), практично кожен раз, коли архівуємо файли, дивимося фотографії, фільми, посилаємо факс або слухаємо музику. Стискаючи файл за алгоритмом Хаффмана перше, що ми повинні зробити, – це

прочитати файл повністю і підрахувати скільки разів зустрічається кожен символ з розширеного набору ASCII, тобто визначити їх відносну частоту у файлі. Якщо ми будемо враховувати усі 256 символів, то не буде різниці в стисненні текстового і EXE файлу.

Моделювання було проведено за допомогою розробленої програми, яка отримує на вхід текстовий документ, проводить кодування по класичному алгоритму Хаффмана (будує кодове дерево та кодує кожен символ) та по його модифікації (розбиває таблицю символів на дві групи згідно їх частот, будує кодове дерево для кожної з них та кодує символи за допомогою них), підраховує розмір стиснених даних за оригінальним алгоритмом та модифікацією. Для проведення експериментів використовувалися дані трьох типів. На першому етапі були використані сгенеровані дані, в яких символи розташовані по убуванню їх частоти, а самі частоти підкоряються закону Ципфа.

На другому етапі були використані сгенеровані дані, в яких розподіл символів також підкоряється закону Ципфа, але розташовані вони були у довільному випадковому порядку. Третій етап експерименту включав дослідження ефективності модифікації алгоритму в порівнянні з класичним на даних, котрі були отримані з реальних баз даних. Базуючись на результатах проведених досліджень першого експерименту можна зробити висновок, що розроблена модифікація алгоритму Хаффмана є більш ефективною в порівнянні з класичним алгоритмом. Для даних, довжина котрих приблизно дорівнює 50 символам, ефективність досягає 30% у порівнянні з класичним алгоритмом. В середньому ефективність досягає 23%. Для цього типу даних доцільно розділяти символи тексту на дві групи, котрі також, приблизно рівні за обсягом. Тестування для реальних даних проводилося для двох наборів даних. Перший набір містив рядки тексту з символів кирилиці довжиною від 10 до 80. Ефективність розробленої модифікації досягає приблизно 13%. Цей показник вірний для даних, котрі суттєво відрізняються довжиною. Найбільший коефіцієнт стиснення отриманий при розділенні символів тексту на дві рівні групи.

Список використаних джерел

1. Vajnberger M., Seroussi G., Schapiro G. The Loco-I stysnennja zobrazhen bez vtrat. Algorytm: pryntsypy i standartyzatsiji v JPEG-LS // Hewlett-Packard Laboratories Technical Report No. HPL-98- 193R1, lystopad 1998, pereroblene zhovtnja +1999. IEEE Trans. Obrobka zobrazhen, Vol. 9 serpnja 2000 roku, pp. 1309–1324

2. Vajnberger M., Seroussi G., Schapiro G. The Loco-I: nyzka skladnist, zasnovana na konteksti, stysnennja zobrazhen bez vtrat Algorytm // Proc. Konferentsija IEEE Data Compression, Snowbird, schtat Juta, berezen-kviten 1996 roku.

3. DISKRETNAJA MATEMATIKA: ALGORITMY. JPEG, JPEG 2000, JPEG-LS. Szhatie izobrazhenij s poterjami i bez [Elektronnyj resurs]: [Veb-sajt]. – Elektronni dani. – Rezhym dostupu: http://rain.ifmo.ru/cat/view.php/theory/data-compression/jpeg-2006.

4. Ватолин Д., Ратушняк А. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео// Д. Ватолин, А. Ратушняк. – М.: ДИАЛОГ-МИФИ, 2002. – 384 с

5. Селомон Д. Сжатие данных, изображений и звука. // Д. Селомон – М.: Техносфера, 2004. – 368 с.

6. Артюшенко В. М., Шелухин О. И. Цифровое сжатие информации // Артюшенко В. М., Шелухин О.И. – М.: Дашков и Ко, 2004. – 426 с.

7. Седжвик Р. Фундаментальные алгоритмы на C++. Части 1-4. Анализ. Структуры данных. Сорттровка. Поиск. // Р. Седжвик Р. – М.: ДиаСофт, 2002. – 688 с.