Plag

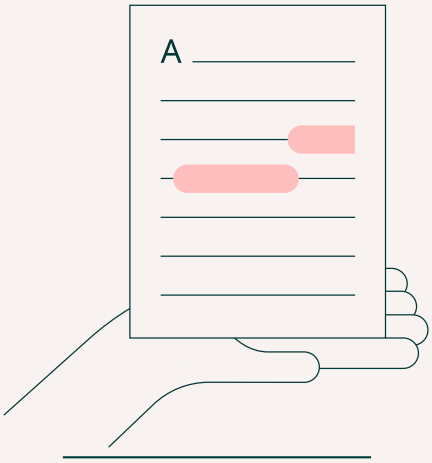
Звіт про оригінальність

● Оцінка схожості 11%

● Ризик плагіату ВИСОКИЙ

Ігор Кагало 2025-06-05 22:45

Посилання на звіт: ZSn3 / Посилання користувача: qfC8



Ось вона –
Ваша звіт про оригінальність!

Ми раді повідомити, що перевірка вашого документа завершена, і результати вже готові! Наші алгоритми старанно працювали, щоб знайти збіги в наших базах даних.

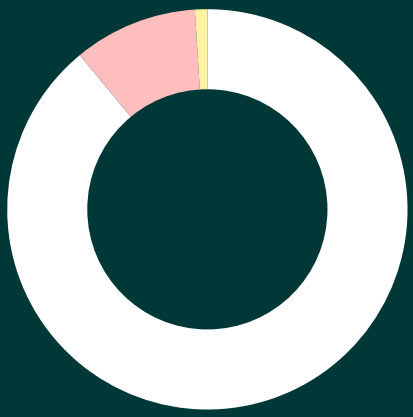
На наступних сторінках ви знайдете результати перевірки:

Бали
Збіги
Посилання

Ваш документ було перевірено за такими джерелами:

- ✓ База даних інтернет-джерел
- ✓ База даних наукових статей
- ○ Глибока перевірка (наш вдосконалений алгоритм)

Бали



Збіги тексту	10%
Перефразування	1%
Цитований текст	0%
Неправильне цитування	0%
Збігів не знайдено	89%

Ризик плагіату ВИСОКИЙ

Ризик плагіату вказує, як збіги тексту розподілені по документу. Вищий ризик виникає, коли збіги з'являються близько один до одного, наприклад, у тому самому абзаці або розділі.

Оцінка схожості 11%

Оцінка схожості показує, скільки слів або символів у вашому документі збігаються з текстами інших документів, включаючи перефразовані тексти або неправильні цитати.

Збіги

1 КОМП’ЮТЕРНИЙ АНАЛІЗ СИМВОЛЬНИХ ДАНИХ

1.1 Аналіз текстів з використанням інформаційних технологій

Текстові редактори дали змогу швидше писати та редагувати книги та статті. Виправляти помилки стало легко, швидко та без надмірних зусиль поміняти місцями фрагменти тексту чи вставити фрагмент тексту з іншого файлу або документа. Великою проблемою було вписувати формули, міняти стиль шрифту тексту, вставка іноземних слів.

Задачі аналізу текстової інформації належать до найбільш поширених задач комп'ютерних інформаційних технологій. Текстові редактори дали змогу швидше писати та редагувати книги та статті. Виправляти помилки стало простіше, можна міняти місцями фрагменти тексту чи вставляти фрагмент тексту з іншого файлу, писати формули, змінювати стилі шрифтів тексту, вставляти іноземні слова і т.д.

Завдяки розвитку комп'ютерної техніки будь-який користувач з мінімальною підготовкою та часовими затратами може підготувати друкований матеріал, а всі графіки, картинки та таблиці розмістити у зручному місці, що також є важливими фактором.

Текст − це послідовність символів, які поділено на рядки. Елементами тестових файлів є символи. У текстах є спеціальні символи, що задають кінці рядків і кінець тексту. Поділ тексту на рядки здійснюється за рахунок особливої інтерпретації спеціальної послідовності символів, яка називається завершуючим символом.

Великим досягненням стало розпізнавання текстів за допомогою сканера. Фрагмент тексту можна сканувати та розпізнавати. Легко можна налагодити контроль грамотності та автоматизувати перевірку синтаксичних помилок. Слово у всіх його формах порівнюється з еталоном, яким служить заздалегідь підготовлений еталонний словник. За допомогою цього словника можна правильно знаходити і виправляти більшість орфографічних помилок, що є дуже практичним. Еталонний словник можна також доповнювати відсутніми словами.

Поява комп'ютерних програм опрацювання текстів сприяла наданню ефективнішої технічної допомогу користувачеві. Можна аналізувати внутрішні характеристики текстів. Незважаючи на деяку суб'єктивність подібного аналізу, він дозволяє побачити твір повністю.

Методи аналізу текстової інформації вимагають обробки великих масивів даних. Це було б неможливо без появи комп'ютерів і подання текстів у цифровому вигляді. З ростом продуктивності комп'ютерів та кількості цифрованих текстів частотний аналіз став доступнішим. За кілька хвилин програма може скласти частотний словник автора і проаналізувати текст за заданою схемою.

1.2 Дослідження текстів з використанням комп'ютерної техніки

Комп'ютерні дослідження текстів беруть свій початок з спроб автоматичного аналізу значних обсягів інформації. Зараз неважко одержати доступ до цифрових версій друкованих засобів. Практично постійно ведеться моніторинг контенту засобів масової інформації. Швидкодії сучасних комп'ютерів цілком вистачає, щоб досить швидко аналізувати будь-які поєднання символів у тексті. Тому неважко витягти з усього цього інформаційного спаму корисну інформацію. Для того тільки потрібно написати програмне забезпечення.

Використання комп'ютерних програм контент-аналізу дозволяє швидко відібрати у великому загальному інформаційному корисну інформацію, яку називають якісним аналізом. Контент-аналіз (англ. 4 content analysis; від content - зміст) − формалізований метод вивчення текстової та графічної інформації, що полягає в перекладі досліджуваної інформації в кількісні показники і її статистичній обробці. Кількісний аналіз дозволяє лише визначити 9 частоту появи в тексті певних характеристик змісту.

Характерною рисою контент-аналізу є висока строгість та систематичність. Суть методу контент-аналізу полягає у фіксації певних одиниць змісту, який вивчається. Цей метод виник із завдань вивчення змісту джерел масової комунікації. Наукова доцільність використання цього методу зумовлюється тим, що він дає змогу отримати й відповідно проаналізувати не окремі факти, а їхню оптимальну сукупність. Структура досліджень включає аналітичні елементи, де об’єктами є окремі джерела, а метою − отримання окремих фактів, і синтетичні, де об’єктом є комплекси джерел, а метою −отримання сукупності фактів.

Метою контент-аналізу вважається не просто оволодіння змістом, але й визначення особистих характеристик автора тексту, його цілей, можливого адресата, зав’язків з подіями в суспільному житті. Для цього досліджується загальний словник матеріалів і різні частоти появи лінгвістичних слів у тексті. Потім визначаються характерні зв'язки між словами, їх змістовність.

Використання комп'ютерного аналізу текстів для їх дослідження сприяє залученню великого числа клієнтів до сучасної техніки. Реальні переваги цифрових технологій проявляються при аналізі дійсно великих масивів інформації, коли з великої кількості документів, необхідно відібрати корисний контент для ретельного якісного дослідження. Тут контент-аналіз виконує функції доброї пошукової машини, що здійснює допоміжну рутинну роботу.

Тому завдяки використанням комп'ютерної техніки зараз вдається спростити або зробити непотрібними багато зайвих операцій опрацювання даних та вибір потрібної інформації. При цьому кількісні методи аналізу текстів відіграють істотно тільки підготовчу роль для подальшої роботи фахівців, що володіють перевіреними методиками якісного дослідження текстів.

1.2 Класифікація функцій для опрацювання символьних даних

Комп’ютерну обробку текстів забезпечують стандартні функції. Це набір різноманітних функцій, призначених для введення символьної інформації з зовнішніх носіїв (клавіатура, жорсткі диски) і виведення її на зовнішні носії (екран, принтер, жорсткі диски), функцій для організації пошуку в текстових даних. На рис. 1.1 наведено класифікацію стандартних функцій для роботи з символьними даними мови програмування С.

Рисунок 1.1–Класифікація стандартних функцій для опрацювання символьних даних

Ці функції знаходяться в бібліотечних файлах string.h і ctype.h мови програмування С і забезпечують швидку реалізацію операцій в процесі опрацювання текстової інформації шляхом її копіювання, доповнення, виділення лексем, порівняння. Використання стандартних функцій дає також можливість перетворювати символьні рядки у числа, а також числа подавати у символьному форматі.

Основну групу становлять функції символьного вводу-виводу. Ряд функцій здійснює операції пошуку в символьних рядках. Ці бібліотечні функції, оголошені в бібліотечному файлі <string.h>.

У багатьох задачах обробки символьної інформації необхідно проводити аналіз символів, які задовольняють певні умови. Функції аналізу символів містяться в бібліотечному файлі <ctype.h>.

1.4 Стандартні функцій для аналізу символів при вводі

У багатьох задачах обробки символьної інформації необхідно проводити аналіз символів, які задовольняють певні умови. Функції аналізу символів містяться в

бібліотечному файлі <ctype.h>.

Практичне значення має задача аналізу символів при вводі. Символи при вводі перевіряються і аналізуються в залежності від того, чи задовольняють вони певні умови. Перевіряється чи є введений символ буквою, цифрою, пустим символом, розділовим знаком або знаком табуляції.

Для аналізу символів при вводі із зовнішніх носіїв використовуються функції, що знаходяться в бібліотечному файлі ctype.h. Файл містить прототипи ряду функцій, які аналізують введені символи. З допомогою цих функцій можна визначити тип введеного символу. Ці функції повертають значення TRUE (не нуль), якщо задовольняється умова символу, що аналізується і FALSE (нуль), якщо введений символ не відповідає заданим критеріям. Аргументом функції є аналізований символ. В таблиці 1.1 наведено перелік функції для аналізу символів при вводі та їх призначення.

Таблиця 1.1 − Функції для аналізу символів при вводі та їх призначення

№ з/п

Прототип функції

Призначення функції

1

int isalnum (int ch)

Повертає TRUE, якщо символ ch — буква або цифра.

2

int isalpha(int ch)

Повертає TRUE, якщо символ ch — буква.

3

int isblank (int ch)

Повертає TRUE, якщо ch пустий символ.

4

int iscntrl (int ch)

Повертає TRUE, якщо символ ch керуючий символ..

5

int isdigit (int ch)

Повертає TRUE, якщо символом ch є тільки цифра

6

int isgraph (int ch)

Повертає TRUE, якщо ch є не пустим символом.

7

int islower (int ch)

Повертає TRUE, якщо символ ch.

8

int isprint (int ch)

Повертає TRUE, якщо символ ch будь-який символ, який відображається.

9

int ispunct (int ch)

Повертає TRUE, якщо символ ch є розділовим знаком.

10

int isspace (int ch)

Повертає TRUE, якщо символ ch є пропуск, знак табуляції, вертикальна табуляція, переклад рядка, прогін сторінки, повернення каретки).

11

int isupper (int ch)

Повертає TRUE, якщо символ ch є буквою верхнього регістру.

12

int isxdigit (int ch)

Повертає TRUE, якщо символ ch є шістнадцятковою цифрою (0-9, a-f, A-F).

На рис. 1.2 показано блок-схему алгоритму аналізу символів при вводі числових даних. за допомогою функції int_get().

Рисунок 1.2 − Блок-схема алгоритму аналізу символів при вводі даних

На рис. 1.3 наведено блок-схему основної функції, яка викликає розроблену власну функцію і виводить введене число, якщо символ цифровий, або “0”, у випадку, коли введений символ не цифра

Рисунок 1.3–Блок-схема основної функції для аналізу символів при вводі даних

Основна функція використовує стандартну функцію isdigit(), яка перевіряє чи є символ цифрою. Якщо символ не є цифрою, то за допомогою функції ungetc() він повертається в потік вводу. алгоритму програми аналізу символів при вводі даних.

Алгоритм враховує всі стандартні символи клавіатури. Розрізняються великі і малі букви латинського алфавіту, цифри, пропуски, розділові знаки. Якщо введений символ співпадає з заданою умовою, то повертається значення TRUE , в противному випадку одержуємо значення FALSE.

2 ВИКОРИСТАННЯ СТАНДАРТНИХ ФУНКЦІЙ ДЛЯ ОБРОБКИ СИМВОЛЬНИХ ДАНИХ

2.1 Класифікація функцій для перетворення рядків символів у числа

1 У багатьох задачах необхідно перетворювати числові дані, записані у формі текстових рядків, в числа 1 різних типів. 1 Такі перетворення реалізують стандартні бібліотечні функції, прототипи яких знаходяться в бібліотечному файлі <stdlib.h>. Перелік функцій для 1 перетворення символьних рядків у числа наведено в табл. 2.1.

Таблиця 2.1–Функції для 1 перетворення символьних рядків у числа

1 № з/п

Функція

Призначення

1

int atoi (char *st);

Функція 1 виділяє у рядку st перше ціле десяткове число і перетворює його у числовий

1 тип 1 int. Числу може передувати довільна кількість пропусків. Кінцем рядка вважається перший символ, що не належить цифрі. Функція повертає шукане число при успішному перетворенн1.

2

long atol (char *st);

Функція аналогічна функції atoi(), але вона перетворює рядок st у число типу long.

3

double atof (char *st);

Функція аналогічна функції 1 atoi(), але перетворює рядок st у число 1 типу 1 double. Число у рядку може бути записане як у 1 цілій так і в дійсній 1 у формі з фіксованою або з плаваючою комою.

4

1 unsigned long strtoul 1 (char *st, char **end, int base);

1 Функція аналогічна функції strtol( ), але вона 1 повертає значення, що має тип unsigned long.

1 5

double strtod (char *st, char **end);

Це розширений варіант функції atof (). Вона 1 перетворює початкову частину рядка 1 st у дане з типом double. 1 Функція додатково повертає через вказівник end адресу першого символу, що записаний за виділеним числом.

6

double strtof (char *st, char **end);*

Функція аналогічна функції strtod (), але вона 1 повертає значення з типом float.

1 7

double strtold (char *st, char **end);*

Функція аналогічна функції strtod( ), 1 але повертає значення типу long double.

2.2 Опис функцій для 1 перетворення рядків символів у числа

1 Для перетворення символьних рядків в цілі числа типу int використовується стандартна функція atoi(), прототип якої має вигляд int atoi (const char * st);

Функція 1 виділяє у рядку st перше ціле десяткове число і перетворює його і 1 ціле число. 1 Числу може передувати довільна кількість пропусків. 1 Кінцем рядка перетворення є 1 перший символ, що не є 1 цифрою. Функція 1 повертає знайдене числове значення при успішному перетворенні. У випадку помилки результат функції не визначений. Якщо в рядку відсутні цифри, то функції повертає “0”. Результати використання функції atoi () наведено в табл. 2.2.

Таблиця 2.2 Перетворення рядків в цілі числа функцією atoi()

Рядок

Значення, повернене функцією atoi()

Пояснення

"190"

190

Ціле число перетворюється повністю

"-2.9"

-2

Пропущені символи ".9", оскільки відбувається перетворення рядка в ціле число. При цьому дробова частина числа відкидається.

"+100y"

100

Функція розпізнає знак +, вважаючи його частиною числа.

"string"

0

Функція atoi() не розуміє слів і бачить тут не число, а лише набір букв.

"x2"

0

Функція atoi() не розуміє літер, яка є першою, і повертає нуль

Для перетворення символьних рядків в цілі числа з типом long використовується стандартна функція atol(), прототип якої знаходиться в бібліотечному файлі <stdlib.h> і має вигляд:

long atol (const char * st);

Функція strtol() – це розширений варіант стандартної функції atol(). Прототип функції знаходиться в бібліотечному файлі <stdlib.h> і має вигляд:

long strtol (const char * st, char **end, int base);

Вказівник end, який є вказівником на вказівник, задає адресу змінної-вказівника, в яку буде записано адресу першого символу, що є не цифрою. Параметр base визначає основу системи числення, в якій записано число. Воно приймає значення від 2 до 36. Цифрами числа є арабські цифри і послідовні малі або великі латинські літери (для системи числення, більшої від десяти). Якщо параметр base рівний 0, то основа системи числення визначається формою його запису. Число, що починається з нуля, відноситься до вісімкової системи числення. Число з префіксом “OX” чи “Ox” належить шістнадцятковій системі числення. Всі решту числа розглядаються як десяткові.

Функція strtol () виділяє з заданого символьного рядка всі цілі числа. Параметрами функції strtol () при її виклику є “strtol (p, &p, 0)”.

Перший параметр p функції strtol () є вказівником на початок поточного числа в рядку str. Другий параметр &p передає у функцію адресу вказівника p, за якою функція запише адресу першого символу, що не є цифрою, що розміщений за знайденим числом. Таким чином, кожен виклик strtol() повертає значення знайденого довгого цілого числа і встановлює вказівник p на символ, з якого треба починати пошук наступного числа.

Для виділення цілих додатних чисел типу “unsigned long” використовується функція strtoul(). Функція strtoul() є аналого функції strtol (), тільки вона повертає значення “unsigned long”. Прототип функції знаходиться в бібліотечному файлі <stdlib.h> і має вигляд:

unsigned long strtoul (const char * st, char **end, int base);

Функція atof() перетворює рядок st в число, яке має тип double. Число у рядку може бути записане в цілій або дійсній формі. Воно може бути як з фіксованою так і плаваючою комою. Прототип функції має вигляд:

double atof (const char * st);

Аргументом функції є вказівник на рядок, який перетворюється. Рядок може мати пропуски перед початком числа та знаки “+” і “-”. Число може складатися з цифр від 0 до 9, десяткової крапки, а також знаку показника степені “e” або “E”. Якщо в рядку відсутні символи для перетворення в дійсне число, то функція повертає “0”. В табл. 2.3 наведено результати використання функції atof().

Таблиця 2.3 Результати перетворення рядків в дійсні числа

Рядок

Значення, яке повертає функція atof()

"12"

12.000000

"-0.123"

-0.123000

"12E+3"

123000.000000

"123.1e-5"

0.001231

Функція strtod() служить розширеним варіантом функції atof(). Прототип функції має вигляд:

double strtod (const char * st, char **end);

Функція 1 перетворює початкову частину рядка 1 st в число 1 з типом double. Вона 1 повертає через вказівник 1 end адресу першого символу, що слідує за виділеним числом.

Функція strtof() є аналогом функції strtod (), тільки вона 1 повертає значення з типом float. Прототип функції має вигляд:

double strtof (const char * st, char **end);

Функція strtold() є аналогом функції strtod(), але вона повертає значення типу long double. Прототип функції має вигляд:

double strtold (const char * st, char **end);

2.3 Опис функцій для перетворення чисел в символьні рядки

В стандартних бібліотеках мови програмування С міститься ряд функцій для зворотних перетворень чисел 2 в символьний рядок. Прототипи цих функцій знаходяться в бібліотечному файлі <stdlib.h> і мають вигляд:

char* 2 itoa (int num, const 2 char* str, int base);

1 char* 2 ltoa (long num, const 2 char* str, int base);

6 char* ultoa (unsigned long num, 2 const 2 char* str, int base);

2 Всі три функції формують із числа num 1 рядок символів, що 2 відповідає запису цього числа в системі числення з основою base. 1 Ці 2 функції відрізняються між собою тільки типом параметра num 1 - цілого числа, яке потрібно перетворити. Функції повертають вказівник 2 на перший символ одержаного рядка. Сформований рядок записується в оперативній пам’яті 1 за адресою str. Функції itoa() та ltoa() 2 записують від'ємні числа 2 зі знаком мінус тільки тоді, 2 коли значення base дорівнює 10. 2 У разі інших основ числення двійкові коди від'ємних 2 чисел 2 розглядаються і перетворюються як беззнакові.

2 Для перетворення дійсного числа або послідовності із декількох чисел 2 в символьний рядок використовується стандартна функція sprintf(). Ця функція аналогічна функції printf (), але вона 2 виконує форматний запис даних 2 у 2 стрінг, адресу якого задає перший параметр функції sprintf().

2 Перетворення дійсного числа 2 в символьний рядок ілюструє наступний приклад:

double zm = 14.7063;

char stzm[30];

sprintf (stzm, "%f", zm);

Функція sprintf() заносить у ділянку пам’яті за адресою stzm символьний рядок "14.7063". Значення zm за специфікацією %f з завершуючим нульовим символом '\0' .

2.4 Функції для пошуку входження символів у рядок

Сучасні програмні засоби для обробки текстів дозволяють здійснювати пошук символів фрагменти текстів. При роботі з текстовими даними функції для пошуку заданих символів і слів в тексті полегшують редагування документів. Для пошуку підрядка в

рядку необхідно перевірити входження заданого слова в даний текст. Якщо цей рядок входить в даний текст, то визначається номер символу тексту з якого виявлено співпадіння. Алгоритми пошуку слів в текстах використовується при індексації сторінок пошуковим роботом, де актуальність інформації в значній мірі залежить від швидкості знаходження ключових слів в текстах.

Бібліотечні файли мови програмування С містять ряд стандартних функцій для організації пошуку в текстах по заданому шаблону. Прототип цих функції знаходиться в бібліотечному файлі string.h, який необхідно підключити за допомогою директиви include .

Для пошуку першого входження заданого символу в рядку використовується функція strchr(). Прототип функції має вигляд:

char * 7 strchr(const char *str, int ch);

7 Функція strchr() виконує пошук першого входження заданого символу ch в рядку, розміщеному в оперативній пам’яті за адресою str. str є адресною константою. Пошук проводиться зліва направо до тих пір, поки не буде виявлено символ ch, або не закінчиться рядок (тобто не зустрінеться завершуючий нульовим символ). При виявленні заданого символу в рядку функція 7 повертає його адресу. Якщо заданий символ відсутній в рядку, то функція повертає адресну константу NULL.

Для обчислення індексу заданого символу в рядку необхідно від значення, яке повертає функція strchr(), відняти значення вказівника str, що є адресою початку рядка. Функція розрізняє регістри символів. Це означає, що велика і мала літери є різними символами. Фрагмент програми ілюструє використання функції strchr() для пошуку символу в рядку.

```
#include <stdio.h>

#include <string.h>

void main(){

char *ptr, buf[80];

int ch;

/* Ввід рядка і символу. */

printf("Enter string buf ");

gets(buf);
```

```
printf("Enter the character ch ");

ch = getchar();

/* Пошук першого входження заданого символу в рядку*/

ptr = strchr(buf, ch);

if ( ptr = = NULL )

printf("Символ %c відсутній.", ch);

else

printf(“character %c \t position %d.\n”, ch, ptr-buf);

}
```

Результат роботи програми:

Рядок asdfghjghhfdsahhgk

Символ h

Результат 5

Для пошуку останнього входження заданого символу в рядок використовується функція strrchr(). Прототип функції має вигляд:

char * strrchr (const 8 char *str, int ch);

8 Функція strchr() виконує пошук останнього входження заданого символу 8 ch в рядку. Пошук проводиться зліва направо до тих пір, поки не буде виявлено останній символ ch, або не закінчиться рядок. При виявленні заданого символу в рядку функція повертає його вказівник. Якщо символ ch відсутній в заданому рядку, то функція повертає константу NULL.

Для обчислення індексу заданого символу в рядку необхідно від значення, яке повертає функція strrchr(), відняти значення вказівника str, що є адресою початку рядка. Функція також розрізняє регістри символів. Це означає, що велика і мала літери є різними символами. Нижче наведено текст програми на мові С, яка ілюструє застосування функції strrchr() для обчислення індексу останнього входження символу в рядок

```
#include <stdio.h>

#include <string.h>
```

```
void main(){

char *ptr, str[80];

int sym;

/* Ввід рядка і символу. */

printf("Enter string str ");

gets(str);

printf("Enter the character sym ");

sym = getchar();

/* Пошук останнього входження заданого символу в рядок */

ptr = strrchr (str, sym);

if ( ptr = = NULL )

printf("Символ %c відсутній.", sym);

else

printf(“character %c \t position %d.\n”, sym, ptr-str);}
```

Результат роботи програми:

Рядок asdfghjghhhhhhhgk

Символ g

Результат 14

2.5 Функція для пошуку позиції входження одного рядка в інший

Для перевірки входження одного рядка в інший використовується стандартна функція strstr(). Прототип функції знаходиться має вигляд:

char * strstr 10 (const char *str1, const char *str2);

10 Функція шукає першу появу одного рядка всередині іншого, причому цілолго рядка, а не його окремих символів. Вона перевіряє, чи входить заданий рядок, розміщений за адресою str2 у рядок, розміщений за адресою strl. Функція повертає вказівник на

перший символ рядка str2 у рядку strl. При відсутності рядка str2 у рядку strl результатом функції є адресна константа NULL.

Якщо рядок для входження str2 має довжину 0, функція повертає вказівник на рядок str1. Відшукавши місце входження одного рядка в інший, обчислюється зсув підрядка str2 відносно початку рядка str1 за допомогою віднімання вказівників. Пошук і порівняння виконуються з врахуванням регістра символів.

Використання функції strstr () для пошуку входження одного рядка в інший проілюстровано наступним фрагментом програми.

/* Пошук в рядку з допомогою функціїstrstr(). */

5 #include <stdio.h>

5 #include <string.h>

5 void main( )

{char *str, str1[80], str2[80];

/* Ввід заданих рядків*/

printf("Введіть рядок для пошуку ");

gets(str1);

printf("Введіть підрядок для пошуку ");

gets(str2);

/* Організація пошуку */

str= strstr(str1, str2);

if (str == NULL)

printf(“Підрядок в рядку не знайдено \n”);

5 else

5 printf("%s was found at position %d.\n", str2, str-str1);

system (“pause”);}

Функція демонструє пошук цілого рядка всередині іншого. Вона повертає вказівник на

першу позицію другого рядка всередині першого або NULL, якщо такого співпадіння не виявлено. Вказівник, який повертає функція, аналізується і виводиться на екран відповідне повідомлення.

2.6 Опис функції для виділення лексем в символьних рядках

Для виділення в одному рядку лексеми, обмеженої символами з другого рядка використовується функція strtok(). Вона повертає вказівник на виділену лексему або константу NULL. Прототип функції має вигляд:

char *strtok (char *str1, const char *str2);

Ця функція виконує поділ символьного рядка str1 на окремі лексеми, записуючи після кожної лексеми нульовий символ кінця рядка '\0'. Рядок str2 задає набір символів, якими мають бути обмежені лексеми рядка str1. Нуль-символ у переліку обмежувачів не вказується.

Для виділення всіх лексем символьного рядка функцію використовують циклічно. У першому зверненні до функції strtok() вказується адреса початку рядка. При цьому функція повертає адресу першої знайденої лексеми. У наступних зверненнях до функції strtok() замість першого параметра записується порожній вказівник NULL, а функція повертає адресу наступної лексеми рядка. Коли всі лексеми виділені, функція повертає NULL.

Нижче наведено програму, яка ілюструє виділення слів-лексем із символьного рядка за допомогою функції strtok().

```
#include <stdio.h>

#include <string.h>

void main ()

{

char str1[] = "Символи, рядки (виділення слів-лексем)";

const char * str2=” ,.;()-“; /* символи-обмежувачі лексем*/

char *pw; /* вказівник на лексеми*/

printf (“\n Слова: \n”);

pw=strtok (str1, str2); /*знаходження першої лексеми*/
```

puts(pw);

pw = strtok (NULL, str2); /* пошук наступної лексеми*/

}

Результати роботи програми:

Результати виконання:

Слова:

Символи

Рядки

Виділення

Слів

Лексем

3 РОЗРОБКА АЛГОРИТМІВ І ПРОГРАМНИХ ЗАСОБІВ ДЛЯ КОМП'ЮТЕРНОЇ ОБРОБКИ ТЕКСТОВИХ ДАНИХ

3.1 Опис програми для аналізу символів при вводі цілих чисел

Розроблена програма PROG_1.c на мові С призначена для аналізу символів при вводі цілих числа з клавіатури або файлу. Програма складається з основної функції void main( ) і функції int get_int(void) для вводу і перевірки цілих чисел. Основна функція складається з наступних кроків:

1 Підключення бібліотечних файлів, в яких знаходяться функції вводу і виводу, а також функції для аналізу символьної інформації.

#include <stdio.h>

#include <ctype.h>

#include <stdlib.h>

#include <string.h>

2 Опис функції для вводу цілих чисел

int get_int(void);

3. Опис файлів для вводу і виводу чисел

```
FILE *fp1,*fp2;

char filename1[30]="d:\\File_getchar.txt";

char filename2[30]="d:\\File_putchar.txt";

char mode_w[4]="w";

char mode_r[4]="r";
```

4 Відкриття файлів.

```
fp1=fopen(filename1,mode_r);

if (fp1!=NULL ) {printf("\nfile %s open mode %s\n",filename1, mode_r); }

else { printf("\nfile %s not open mode %s\n", filename1, mode_r); exit(1);}

/* Відкриття файлу File_putchar.txt, який містить послідовність бітів */

fp2=fopen(filename2,mode_w);

if (fp2!=NULL ) {printf("\nfile %s open mode %s\n\n",filename2, mode_w); }

else { printf("\nfile %s not open mode 3 %s\n\n", filename2, mode_w); exit(2);}
```

5 Оголошення цілої змінної x, якій присвоюється значення функції get_int(), виклик функції get_int(), вивід значення функції на екран та запис у файл.

```
int x;

x = get_int();

if (x!='*')

{

printf("ENTERED SYMVOL --- %d\n", x);

fprintf(fp2,"ENTERED SYMVOL --- %d\n", x);}

else { printf("ENTERED SYMVOL --- %c\n", '*');

fprintf(fp2,"ENTERED SYMVOL --- %c\n", '*');
```

}

Функції int get_int(void) призначена для аналізу символів при вводі цілих числа з клавіатури або файлу. Програма пропускає пусті символи. Якщо введений непустий символ є недопустимим в числі, то повертається символ «*» і виводиться повідомлення про помилку. Функція складається з наступних кроків:

1 Читання символу ch функцією fgetc(fp1) з файлу, аналіз його функцією isspace(). Якщо ch належить до пустих символів, то вводиться наступний символ. Ввід продовжується до тих пір, поки в потоці не з'явиться непустий символ.

2 Перевірка, чи дозволений даний символ в записі числа. Якщо символ не є знаком мінус, плюс, цифрою або кінцем файлу,

if (ch != '-' && ch != '+' && !isdigit(ch) && ch != EOF)

то за допомогою функції ungetc () символ поміщається назад в потік введення, і функція повертає управління в головну функцію main().

ungetc(ch, stdin);/* Функція повернення символу в потік вводу */

3 Саме повернений символ буде першим, який програма введе при наступній операції вводу з даного потоку. Це необхідно тоді, коли функція get_int () вводить нечисловий символ з потоку stdin, і вона повинна повернути його назад.

4 Якщо символ належить до тих, які можуть використовуватися при записі цілих чисел, то функція продовжує роботу.

5 Якщо введений символ має знак мінус, то встановлюється відповідний знак числа:

if (ch == '-') sign = -1;

6 Обробляється знак числа. Якщо був введений знак мінус, то змінна sign встановлюється рівною -1. Оскільки числа можуть бути від’ємними, то після введення знаку «мінус» введення числа продовжується. Якщо був введений знак мінус, то необхідно ввести наступний символ з потоку:

if (ch == '+' || ch == '-') ch = getchar();

7 Символи вводяться підряд один за одним до тих пір, поки є цифрою:

for (i = 0; isdigit(ch); ch = getchar())

i = 10 * i + (ch - '0');

8 Ввід закінчується при появі нецифрового символу. Міняється знак для від’ємного числа:

i = i*sign;

9 Якщо останній введений символ не є символом кінця файлу, то його необхідно повернути в потік введення.

if (ch != EOF) ungetc(ch, stdin);

Результатом роботи програми є введений символ і його повторення, якщо він числовий. При вводі не числового символу програма виводить повідомлення. Повний текст програми на мові С наведено в додатку А.

На рисунку 3.1 показано вмістиме файлу File_getchar.txt вхідних даних, коли всі символи є цифрами.

Рисунок 3.1 − Вмістиме файлу File_getchar.txt з цілими числами

На рисунку 3.2 показано результат виводу цілих чисел.

Рисунок 3.2 − Результат виводу цілих чисел

На рисунку 3.3 показано вмістиме файлу File_getchar.txt вхідних даних, коли не всі символи є цифрами.

Рисунок 3.3 − Вмістиме файлу File_getchar.txt з цифрами та іншими не числовими символами

На рисунку 3.4 показано результат виводу тільки числових символів.

Рисунок 3.4 − Результат виводу цілих чисел без інших символів

На рисунку 3.5 показано вмістиме файлу File_getchar.txt вхідних даних, коли перший символ не є цифрою.

Рисунок 3.5 − Вмістиме файлу File_getchar.txt з першим не числовим символом

На рисунку 3.6 показано результат виводу символу «*».

Рисунок 3.6 − Результат виводу символу «*»

3.2 Опис програми для перетворення рядків символів в числа

Розроблена в дипломному проєкті програмний код PROG_1.c на мові С демонструє

процес перетворення символьних рядків в числа різних типів. Програма переводить символьний рядок, введений з клавіатури, в число, тип якого також задається параметром, введеним з клавіатури.

Програма PROG_1.c призначена для перетворення рядків символів в цілі числа типу int і long та в дійсні числа типу float. Для перетворення символів в цілі числа використовуються функції відповідно atoi() і atol(). Для перетворення символів в дійсні числа використовуються функція atof().

Програма працює в інтерактивному режимі. З клавіатури вводиться рядок, призначений для перетворення в число. Режим перетворення задається змінною p, що вводиться з клавіатури.

Якщо p =1, рядок перетворюється в ціле число типу int.

Якщо p =2 рядок перетворюється в ціле число типу long.

Якщо p =3 рядок перетворюється в ціле число типу float.

При введенні значення p, відмінного від наведених вище значень, програма видає повідомлення про помилку і очікує введення потрібного значення p. Програма закінчує роботу при введенні пустого рядка.

Програма складається з наступних кроків.

1 Підключення бібліотечних файлів, які містять прототипи стандартних функцій файлового вводу-виводу, функцій обробки символьної інформації та функцій роботи системи:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

2 Опис змінних:

char data[255];

int i;

long l;

float f;

3 Введення рядка для перетворення.

printf("\nEnter the string to convert ('0' to exit):\n\n ");

scanf("%s", data);

4 Перевірка введеного рядка. При введенні пустого рядка програма закінчує роботу.

5 Ввід значення змінної p для індикації типу числа, в який перетворюється рядок:

printf("Vvedit parameter p\n");

printf("\n p=1 - int, p=2 - long, p=3 - float \n\n p=");

scanf("%d", &p);

6 Застосування оператора розгалуження для вибору значення p:

При p=1 викликається функція atoi( ), яка перетворить введений рядок data в ціле число.

При p=2 викликається функція atol( ), яка перетворить введений рядок data в ціле число типу long.

При p=3 викликається функція atof(), яка перетворить введений рядок data в дійсне число і присвоює його змінній f типу double.

7 При введенні значення для змінної p, відмінного від вище перерахованих варіантів, виводиться повідомлення про помилку і запрошення вводити нове значення p.

Повний текст програми для перетворення рядків символів у числа різних типів наведено в додатку Б. Результат роботи програми перетворення символьних рядків в числа показано на рис. 3.7.

Рисунок 3.7 − Результат роботи програми перетворенні рядків в числа

Результат роботи програми при введенні не числових символів показано на рис. 3.8.

Рисунок 3.8 − Результат роботи програми при введенні не числових символів

3.3 Опис алгоритму і програми для пошуку і заміна слів в тексті

При комп’ютерній обробці текстів часом виникає потреба для пошуку групи символів, слів, фрагменту тексту та їх заміни іншими символами. Для пошуку і заміни в мовах програмування існує ряд стандартних функцій.

З Функції пошуку та заміни здебільшого застосовуються до символьних рядків у текстових файлах. Вони З економлять час пошуку З певного слова чи рядка у тексті документа вручну, а також перезапис його вмісту, тобто заміну. Можна здійснювати пошук певних слів, фраз, чисел і символів та автоматично замінювати результати пошуку новим указаним вами вмістом. До обсягу пошуку входить увесь видимий вміст документа, як-от основна частина тексту, колонтитули, таблиці, текстові поля, фігури, виноски та коментарі.

Для пошуку слова в тексті в мові С використовується стандартна функція strstr(). Функція перевіряє, чи задане слово входить в текст. Якщо задане слова знайдено, то функція повертає адресу першого символу входження слова в текст. Якщо шукане слово відсутнє, то функція повертає константу NULL. При знайденому слові, за необхідності, можна зробити відповідну заміну. З Під час пошуку або заміни одного слова можна перейти до наступного знайденого слова і його заміни.

Алгоритм для пошуку і заміни слів в тексті складається з наступних елементів:

Ввід тексту для опрацювання;

Ввід слова для пошуку:

Ввід слова для заміни:

Пошук слова за заданим ключем за допомогою функції strstr();

Вивід адреси знайденого слова в тексті.

Заміна шуканого слова за знайденою адресою.

Структурну схему алгоритму для пошуку слів в тексті і їх заміну показано на рис. 3.9.

Рисунок 3.9 − Структурна схема алгоритму для пошуку і заміни слів в тексті

На основі алгоритму розроблено програму PROG_3.c. Розроблена програма демонструє пошук заданих слів в текстовому файлі. Текст для пошуку знаходиться в файлі File_getchartext.txt в текстовому форматі. Слово word1 для пошуку і слово word2 для заміни вводиться з клавіатури. Програма складається з наступних кроків:

1. Підключення бібліотечних файлів, які містять прототипи стандартних функцій файлового вводу-виводу, функцій обробки символьної інформації та функцій роботи системи:

#include <stdio.h>

#include <stdlib.h>

#include <stdring.h>

2. Опис вказівника на змінну структурного типу FILE, який асоціює файл на диску з потоком вводу, відкриття файлу.

FILE *fp;

fp=fopen("d:\\File_getchartext.txt","r");

3. Опис тексту і слів для пошуку та заміни.

char text[256]; /* Тексто для пошуку */

char word1[10]; /* Слово для пошуку */

char word2[10];/*Слово для заміни*/

4. Читання тексту для пошуку з файлу за допомогою функції fgets( ). Текст поміщається в ділянку пам’яті з адресою text.

printf("Enter the string to be searched: \n");

fgets(text, 256,fp);

5. Ввід слів для пошуку та заміни.

printf("Enter the word1: \n");

gets(word1);

printf("\nEnter the word2: \n") ;

gets(word2);

6. Для контролю текст і слова для заміни та пошуку виводяться на екран.

puts(text);

puts(word1);

puts(word2);

7 Пошук заданого слово для заміни за допомогою функції strstr(). Заміна здійснюється оператором *(ptr+j)=*(word2+j).

```
for(i=0; i<strlen(text)-k;i++) {

ptr=strstr(ptr,word1);

// printf("\nptr=%lld\n", ptr);

if(ptr==NULL)

{printf("\nptr=%lld\n", ptr);

break;}

for(j=0; j<k;j++)

*(ptr+j)=*(word2+j); }
```

8. Вивід результатуючого тексту на екран.

```
puts(text);
```

Якщо шукане слово відсутнє, то функція ptr=strstr() повертає вказівник NULL і програма закінчує роботу.

На рисунку 3.10 наведено результати роботи програми PROG_3.c.

Рисунок 3.10 − Результати роботи програми PROG_3.c

В початковому тексті слова “for” замінюються словами “FOR”, а слова «strlen» замінюються словами «STRSTR».

На рисунку 3.11 наведено результати роботи програми PROG_3.c у випадку відсутності шуканого слова для заміни. Тоді функція ptr=strstr() повертає вказівник NULL і програма закінчує роботу.

Рисунок 3.11 − Результати роботи програми PROG_3.c у випадку відсутності шуканого слова для заміни

Повний текст програми наведено в додатку В.

Посилання

Це джерела виділених збігів у вашому документі. Кожен збіг позначено темно-зеленим числом, яке відповідає вказаному тут джерелу. Джерела впорядковані за схожістю — чим вищий бал, тим сильніше збіг.

#	Джерело	%
1	studfile.net	5.5%
2	pdf.lib.vntu.edu.ua	2.1%
3	uk.wikipedia.org / Пошук_і_заміна	0.8%
4	elibrary.kubg.edu.ua	0.6%
5	comp.nus.edu.sg	0.2%
6	dspace.tul.cz	0.2%
7	eir.zp.edu.ua	0.2%
8	crust.ust.edu.ua	0.2%
9	reposit.uni-sport.edu.ua	0.1%
10	documentation.help	0.1%

Plag

Дякуємо, що перевірили
свій документ за допомогою
Plag!