Обробка тексту

Функції обробки тексту. По символьна обробка тексту. Пошук заданого підрядка в тексті. Алгоритм Бойєра-Мура. Використання хеш-функції для пошуку довільного підрядка у рядку. Рекурсивний синтаксичний аналіз виразів із дужками.

Рядкові величини

Pascal (Delphi)

- операція + (з'єднання) 'місто'+' '+'Луцьк'

Функції роботи з рядками:

№

Назва функції

Призначення

Приклад

Результат

1.

Length(S)

визначає кількість символів у заданому рядку

Length (‘місто Луцьк’)

11

2.

Copy(S,n,m)

виділяє m символів рядка S, починаючи від символу з номером n

Copy (‘місто Луцьк’, 6, 5)

‘Луцьк’

3.

Pos(S1, S2)

визначає номер символу, з якого починається входження рядка (тексту) S1 у рядок S2

Pos (‘ ‘,‘місто Луцьк’)

6

4.

Concat(S1, S2,...)

з'єднує рядки в один рядок

Concat('20', '01')

‘2001’

Процедури роботи з рядками:

№

Назва функції

Призначення

Приклад

Результат

1.

Insert (A:string, var B: string, n:integer)

вставляє рядок A у рядок B, починаючи від позиції з номером n

S1:=’місто’;

S2:=’Луцьк’;

Insert(S1,S2,1);

’містоЛуцьк’;

2.

Delete (var S:string, n:integer, m:integer)

вилучає m символів з рядка S, починаючи від позиції n

S:=’містоЛуцьк’;

delete(S,1,5);

’Луцьк’;

3.

Str (A:integer, var S:string)

переводить числове дане A у дане типу рядок

A:=2001;

Str(A,S);

‘2001’

4.

Val (S: string, var A, KOD: integer)

засилає у числову змінну A числовий образ рядка S, повертаючи код помилки KOD

S:=’2001’;

Val(S,A,Kod);

2001

C++

Функції для опрацювання рядків

Модуль string.h

strlen(<рядок>) - визначає фактичну кількість символів у рядку, застосовується у виразах;

strcat(rl, r2) - команда з'єднання рядків г1, г2 в один ря-док, результат присвоює змінній г1;

strncat(M, г2, п) - до змінної г1 додає перших n символів рядка г2, команда;

strcpy(r1, r2) - копіює символи з рядка г2 в рядок г1, команда;

strncpy(r1, r2, n) - копіює перших n символів рядка г2 в рядок r1, команда;

strchr(r1, <символ>) - визначає перше входження деякого символу у рядок r1 так: повертає рядок, який починається від першого входження заданого символу до кінця рядка r1, застосовується у виразах;

strrchr(r1, <символ>) - визначає останнє входження зада-ного символу у рядок, застосовується у виразах;

strspn(r1, r2) - визначає номер першого символу, який входить у рядок г1, але не входить

у рядок г2, застосовується у виразах

strstr(r1, r2) - визначає в рядку г1 підрядок, що починаєть-ся з першого входження рядка г2 у рядок г1, засто-совується у виразах;

strtok(r1, r2) - визначає частину рядка r1, яка закінчується перед першим однаковим символом рядків r1 та r2;

strnset(r1, <символ>, n) - вставляє n разів заданий символ • перед рядком M, застосовується у виразах;

strupr(rl) - перетворює усі малі літери рядка у великі;

strlwr(rt) - перетворює усі великі літери рядка у малі;

strrev(rl) - записує рядок у зворотному порядку.

Застосування функцій

Результат

Lviv = "НУ Львівська політехніка"

n = strlen(Lviv)

n = 21

strcat(Un, Lviv)

Un = "НУ Львівська політехніка"

strncat(Un, Lviv, 10)

r1 = "НУ Львівська"

strcpy(r1, Lviv)

r1 = "Львівська Політехніка"

strncpy(r1, Lviv, 10)

r1 = "Львівська"

p = strchr(Lviv, 'П')

p = "політехніка"

p = strrchr(Lviv, Ї)

p = "іка"

n = strspn(Lviv, "Львів")

n = 5

p = strstr(Lviv, "теж")

p = "техніка"

p = strtok(Lviv, "кс")

p = "Львів"

p = strnset(Lviv, 'x', 10)

p = "xxxxxxxxxxполітехніка"

```
p = strupr("I Love You")
```

```
p = "i love you"
```

```
p = strlwr("I Love You")
```

```
p = "I LOVE YOU"
```

```
p = strrev('техніка")
```

```
p = "акінхет"
```

Задачі

Задача1. **ACM World Finals**

Ім’я вхідного файлу: acm.in

Ім’я вихідного файлу: acm.out

Дехто з вас, мабуть, знає, що кожного року проводиться чемпіонат світу з програмування серед студентів. У фінал цього змагання проходять близько 80 команд з усього світу.

Кожна команда складається з трьох чоловік і має назву. Напишіть програму, яка по короткій назві команди і прізвищах її учасників, формує повну назву команди.

Повна назва команди складається з короткої назви команди і списку прізвищ її учасників. Прізвища учасників у списку мають бути впорядковані за абеткою і відділені одне від іншого комами. Назва команди від прізвищ учасників має бути відділена двокрапкою. Після кожного розділового знаку має бути рівно один пробіл.

Формат вхідних даних: вхідний файл містить рівно 4 рядки. Перший рядок містить назву команди. Кожен із наступних трьох рядків містить прізвище одного із членів команди. Довжини рядків не перевищують 50 символів.

Формат вихідних даних: єдиний рядок вихідного файлу має містити рівно один рядок з повною назвою команди.

Приклад вхідних і вихідних даних:

acm.in

acm.out

Dream Team

Knuth

Dijkstra

Cormen

Dream Team: Cormen, Dijkstra, Knuth

Задача2. GoTo.

Учні, недавно що почали програмувати, вживають дуже багато операторів GOTO, що є майже неприпустимим для структурованої програми. Допоможіть викладачу інформатики написати програму, яка оцінюватиме ступінь структурованості відладженої програми школяра на мові Pascal, спершу просто підраховувавши кількість операторів GOTO в ній.

В синтаксично вірній програмі ключове слово оператора переходу GOTO може стояти або на початку рядка або після пропуску або після одного з символів — ‘;’, ‘:’, ‘}’, а після нього може стояти або пропуск або переклад рядка або символ ‘{‘ (табуляцію як роздільник розглядати не будемо).

Нагадаємо, що окрім позначення дійсно оператора переходу, слово GOTO може зустрічатися в тексті програми в рядкових константах, укладених в апострофи, або в коментарях, причому для простоти вважатимемо, що коментар завжди починається з символу ‘{‘, а закінчується першим що зустрівся після цього символом ‘}’, при цьому символ ‘{‘ повинен знаходитися не усередині рядкової константи. В цих випадках слово GOTO підраховувати не потрібно. Рядкові і прописні букви в Pascal не помітні.

У вхідному файлі goto.in знаходиться текст програми без синтаксичних помилок на мові Pascal. Розмір програми не перевершує 64K. У вихідному файлі goto.out повинне виявитися одне число – кількість операторів GOTO в цій програмі.

Приклад вхідного файлу:

```
label 1,2;

var I:byte;

begin

 1: I:=I+1;

 if I>10 then goto 2 else

 writeln(¢ goto ¢); GoTo 1;

 2: if I<10 then goto 1 {else { goto 2;}

end.
```

Вихідний файл для наведеного приклад

3

Задача 3. *Дужки.*

Перевірити в виразі правильність розставлення дужок. Вивести повідомлення (Yes|No).

Задача 4. Вираз

Обчислити вираз, який містить операції(+,-,,/), цілі та дійсні числа (2, 2.5, -5.02), дужки.*

Тема: Синтаксичний розбір виразів

Однієї з головних причин, що лежать в основі появи мов програмування високого рівня, з'явилися обчислювальні задачі, що вимагають великих об'ємів рутинних обчислень. Тому до мов програмування пред'являлися вимоги максимального наближення форми запису обчислень до природної мови математики. В зв'язка з цей одній з перших областей системного програмування сформувалося дослідження способів виразів. Тут отримані численні результати, проте найбільше розповсюдження отримав метод трансляції за допомогою зворотного польського запису, який запропонував польський математик Я. Лукашевіч.

Нагадаємо, що існує цілий ряд методів синтаксичного розбору і обчислення виразів. Нехай вирази є рекурсивними структурними даними, які визначаються в термінах самих себе. Якщо ви обмежитеся використанням у виразах тільки операторів +, - * / і дужок , то зможете сказати, що всі вирази можуть бути визначеними в термінах наступних правил:

выраз=>терм[+терм][-терм]

терм=>множник[*множник][/множник]

множник=>змінна, число або (вираз)

де будь-яка частина може бути порожньою. Квадратні дужки позначають необов'язковість, а => - утворення. Фактично правила є правилами утворення виразів.

Вираз 10+5*B

складається з двох термів: 10 і 5*B. Проте, включає три множники: 10, 5 і B. Цими множниками є два числа і одна змінна.

З другого боку вираз

14*(7-C)

має два терми 10 і (7-C), один з яких є числом, а інша дочірнім виразом. Дочірній вираз розпадається на одне число і одну змінну.

Даний процес формує основу для рекурсивного низхідного синтаксичного розбору, який є набором загальних рекурсивних процедур, що носять характер ланцюжка. На кожному відповідному кроці синтаксичний розбір може виконувати задані операції в алгебра правильній послідовності. Наприклад розглянемо синтаксичний розбір вхідного виразу 9/3-(100 +56) і виконання операцій по кроках.

Крок 1. Узяти першу лексему: 9/3

Крок 2. Узяти обидва множники і виконати операцію поділу.

В результаті виходить 3.

Крок 3. Узяти другу лексему: (100+56). В даній точці ви повинні рекурсивно проаналізувати другий вираз.

Крок 4. Узяти обидва числа і скласти. В результаті виходить 156.

Крок 5. Повернутися з рекурсивного виклику і відняти 156 з 3, що дає відповідь - 153.

Якщо ви трохи заплуталися, не турбуйтеся. Це складна концепція. Потрібно засвоїти два моменти про даний рекурсивний погляд на вирази: по-перше, передування операторів є неявним при заданих правилах породження виразів; по-друге, даний метод синтаксичного розбору і обчислення виразів дуже схожий на те, як це робиться уручну.