ALGORITHMS
ZELLER'S CONGRUENCE
AN IN-PLACE SORT
BUILDING A NEURAL NET
SUBSTRING SEARCHES IN C

REVIEWS
CHOOSING A PC ADA
SLR SYSTEMS' OPTASM
SOLID B+ TOOLKIT
Contents

December 1988

Articles

34 Day of the Week Revisited
Irving Sperling. Did Columbus discover the Bahamas on a Tuesday or a Wednesday? Were the Angles and the Jutes surprised at work when the Normans invaded England in 1066, or were they enjoying a relaxing weekend at home? With the Zeller day-of-the-week algorithm, you'll know for sure.

41 Sorting in Place
K. O'Toole. Sometimes algorithms result from the workings of a devious mind...as in the case of this routine for sorting data elements without temporary storage buffers or arrays of pointers.

53 The Polynomial ADALINE Algorithm
Maureen Caudill. First there was ADALINE, a neural network for classifying linearly separable elements. Now there is PADALINE, for advanced pattern-matching applications.

63 String Searching in C
Sanford J. Hersh. C's lack of a Pos function has occasionally led to some creative coding tricks. This APL-style algorithm locates substrings and reports their location in a truth table vector.

Product Reviews

89 Choosing a PC Ada
98 SLR Systems' Optasm 1.5
103 Solid Software Inc.'s B+ Toolkit

Departments

9 Feedback
17 Programming on Purpose
P.J. Plauger. Protecting intellectual property
25 Tools of the Trade
Warren Knuffel. Creating executable specifications with MicroSTEP
67 Bit by Bit
Stan Kelly-Bootle. Let us now exploit famous persons
73 Designing with Databases
Matthew Rapaport. Text-based database management systems
82 Product Bingo
87 Public Domain Software Review
Tim Parker. SNAP! Updated
111 Annual Index
112 Advertiser Index
116 Classified Connection
119 EOF
T.A. Elkins. A modest proposal
String Searching

in C

This APL-style algorithm
locates substrings within character strings

Sanford J. Hersh

C is a language that should make string manipulation easy. Unfortunately, it typically does not include commands to do substring searching. How many occurrences of a substring are within a string and where are they? This article presents an algorithm that addresses this question.

This routine (Listing 1) is based somewhat on the technique that would be used to solve the same problem in APL (Listing 2). The major difference is that in APL, all tests are made first to produce a truth table matrix, which is then shifted and compressed to a vector. In this C routine, a truth table vector is built. Each succeeding test result is shifted and then overlays the vector. In both cases, the resulting vector is used to produce position integers.

The function STRSRCH returns a pointer to an integer vector that is terminated by a negative one. Any positive integer before this terminator is a position of the substring within the string. To find out how often this substring occurs within the string, these numbers simply need to be counted. If any negative numbers appear before the terminating negative one, a failure occurred during boundary checking.

The program starts with five if statements that do nothing more than make boundary checks. The heart of the algorithm is contained within the next three for loops. The first one creates the initial truth table vector. The second overlays all the others over it. The third converts this vector into positions.

The maximum number of positions is arbitrarily defined by sizvec. This number can be changed at the whim of the programmer. Listing 3 (STRTEST) is an example of how this program is used.

Sanford Hersh is the principal of COMPUTATIONS, a company that specializes in problem-solving in manufacturing, warehousing, and inventory control. He has a B.S. in electrical engineering from Rensselaer Polytechnic Institute, Troy, N.Y., and an M.S. in systems and information science from Syracuse University, Syracuse, N.Y.

Artwork: Dwight Been
listing 1. (continued on following page)

/* This C program is based on the following APL program: */

/*
[0] VEC #is STRING STRSRCH SUBSTG;#0;ARY
[1] (* returns all positions of SUBSTG within STRING
[2]   #0 = #is 0
[3]   STRING #is STRING & SUBSTG #is SUBSTG
[4]   ARY #is #and(#isota #rho SUBSTG)#theta STRING #outer#SUBSTG
[5]   VEC #is ARY/#isota #rho STRING

$ */

/* STRSRCH returns a pointer to an integer vector that terminates with -1.*/

Any positive number preceding the -1 is a position of the substg
within the string. A negative number preceding the -1 represents an error. */

typedef char * char_ptr;
#define szvec 15
#define NOMEM ( char_ptr * )

int *strsrch( string, substg )
char_ptr string, substg;

{ register int x, y;
  static int vec[ szvec + 1 ]; /* leave room for -1 */
  unsigned int sizstg, sizsub, endsiz. *ary;
  char_ptr startmem. malloc( );
  int strlen( );

  sizstg = strlen( string );
  sizsub = strlen( substg );
  endsiz = sizstg - sizsub + 1;

  /* Check boundary conditions: */

  if ( ( sizstg == 0 ) && ( sizsub == 0 ) )
    { vec[ 0 ] = -5;
      vec[ 1 ] = -1;
      return( vec );
    }

  if ( sizsub == 0 )
    { vec[ 0 ] = -3;
      vec[ 1 ] = -1;
      return( vec );
    }

  if ( sizstg == 0 )
    { vec[ 0 ] = -2;
      vec[ 1 ] = -1;
      return( vec );
    }

  if ( sizsub > sizstg )
    { vec[ 0 ] = -6;
      vec[ 1 ] = -1;
      return( vec );
    }

  return( vec );
}
LISTING 1. (Continued from preceding page)

```c
if ( NOMEM == ( ary = startmem = malloc( endsiz * sizeof( int ) ) ) )
{
    vec[ 0 ] = -9;
    vec[ 1 ] = -1;
    return( vec );
}
/* Start of algorithm: */

for( x = 0, x ( endsiz, x++ ) )
    *ary++ = string[ x ] == substg[ 0 ];
for( y = 1, ary = startmem, y ( sizsub: y++, ary = startmem )
    for( x = y, x ( endsiz, y ), x++ )
        *ary++ & string[ x ] == substg[ y ];
for( y = 0, x = 0, ( x ( endsiz ) & ( y ( sizvec ), x++ )
    if ( *ary++)
        vec[ y++] = x;
vec[ y ] = -1;
free( startmem );
return( vec );
```

LISTING 2.

```
VSTESRC(\[]\[]\[]\[]\[]\[]
[0] VEC+STRING STRSRC SUBSTG;\0;ARY
[1] \*RETURNS ALL POSITIONS OF <SUBSTG> WITHIN <STG>
[2] \*\0=0
[3] STRING+,STRING \* SUBSTG+,SUBSTG
[4] ARY++/(\1\p\SUBSTG)\*STRING+,=SUBSTG
[5] VEC+ARY;=STRING
```

LISTING 3.

```
/* This function displays: */
/* The positions are: */

#include ( stdio.h)
extern int *strsrc( );

int main( )
{
    static char *stg1 = "The files are located in the file cabinet.
    Please file this memo in the circular file!";
    static char *stg2 = "file";
    int *ptr;
    ptr = strsrc( stg1, stg2 );
    printf( "The positions are: \n" );
    while( -1 != *ptr )
        printf( "\n\t", *ptr++ );
    printf( "\n" );
    return( 0 );
}
```