

COMPUTER LANGUAGE™

VOLUME 5, NUMBER 12

DECEMBER 1988

\$3.50 Canada \$4.50

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



COMPUTER
LANGUAGE

Cover Art: Did you know that "algorithm" is derived from al-Khuwarizmi, the name of a ninth-century Arabic mathematician? And "ladder" comes from the Latin *clinare* ("to lean"), with stops along the way in Greek, Old High German, and Old English.

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 Candill. First there was ADALINE, a neural network for classifying linearly separable elements. Now there is PADATALINE, for advanced pattern-matching applications.
- 63 String Searching in C**
Sanford J. Hersb. 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. Planger. Protecting intellectual property
- 25 Tools of the Trade**
Warren Kenffel. 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

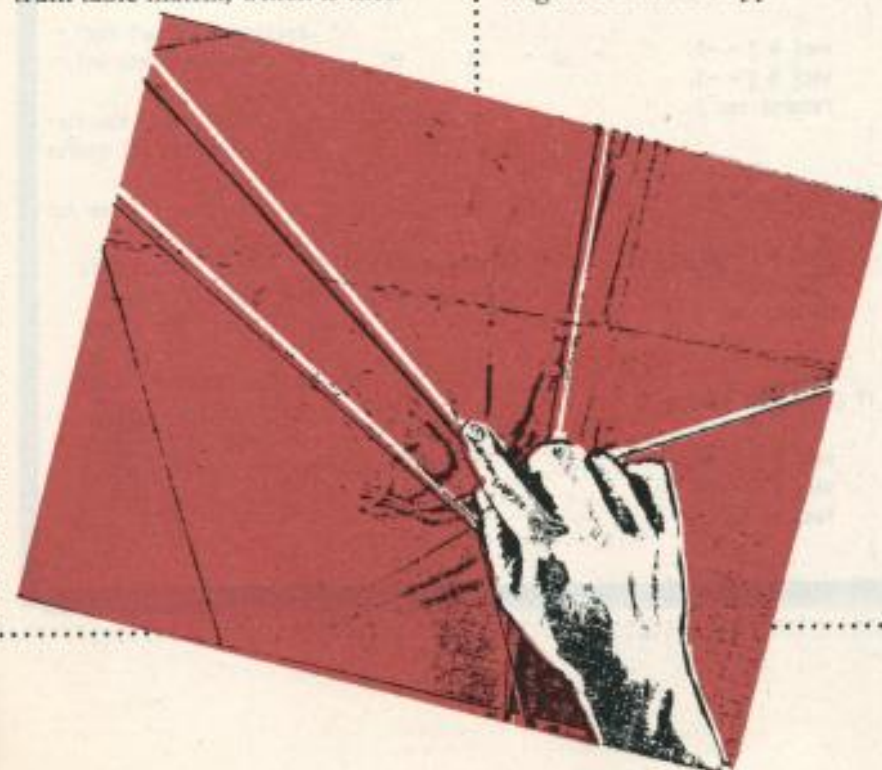
the terminating negative one, a failure occurred during boundary checking.

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 *sizevec*. 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 COM-PULATIONS, 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 Beem



Eco-C88 C Compiler with Cmore Debugger

Professionals prefer the Eco-C88 C compiler for ease of use and its powerful debugging features. Our "picky flag" gives you nine levels of lint-like error checking and makes debugging easy:

"I'm very impressed with the compiler, editor, and debugger. I've tried quite a few different compilers for the PC and have given up on all of the others in favor of yours... I've gotten to the point where I download C code from a DEC VAX/VMS system just to be able to compile it with the picky flag set at 9. It finds lots of things VMS totally ignores..."

JS, Oak Ridge, TN

The Eco-C88 compiler includes:

- A full-featured C compiler with 4 memory models (up to 1 meg of code and data) plus most ANSI enhancements.
- Without a doubt, the best error checking you can get. We catch bugs the others miss, making you much more productive.
- Cmore is a full-featured source code debugger, not some stripped-down version.
- Robust standard library with over 230 useful (no "fluff") functions, many of which are System V and ANSI compatible. Full source is available for only \$25.00 at time of order.
- CED, a fast, full screen, multiple-window program editor with on-line function help. You can compile, edit, and link from within CED.
- cc and mini-make utilities included that simplifies the most complex compiles.
- Users manual with over 150 program examples (not fragments) to illustrate how to use the library functions.
- Fast compiles producing fast code.

Our Guarantee: Try the Eco-C88 compiler for \$99.95. Use it for 30 days and if you are not completely satisfied, simply return it for a full refund. We are confident that once you've tried Eco-C88, you'll never use anything else. Call or write today!

Orders: **1-800-952-0472**

Info: **1-317-255-6476**



Ecosoft Inc.

6413 N. College Avenue
Indianapolis, IN 46220

ECOSOFT

Circle No. 33 on Inquiry Card

String Searching in C

LISTING 1. (Continued on following page)

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

/* $
[0] VEC #is STRING STRSRCH SUBSTG:#io:ARY
[1] | returns all positions of SUBSTG within STRING
[2] #io #is 0
[3] STRING #is STRING & SUBSTG #is SUBSTG
[4] ARY #is #and/(#iota #rho SUBSTG)#theta STRING #outer=SUBSTG
[5] VEC #is ARY/#iota #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 sizvec 15
#define NOMEM ( char_ptr )0

int *strsrch( string, substg )
char_ptr string, substg;
{
    register int x, y;
    static int vec[ sizvec + 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 );
    }
}
```



LISTING 1. (Continued from preceding page)

```
if ( NOEMEM == ( 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.

```

VSTRSRCH[ ]V
[0] VEC+STRING STRSRCH SUBSTG;IO;ARY
[1] * RETURNS ALL POSITIONS OF <SUBSTG> WITHIN <STG>
[2] IO+0
[3] STRING+,STRING < SUBSTG+,SUBSTG
[4] ARY+^/( \pSUBSTG)@STRING.=SUBSTG
[5] VEC+ARY/\pSTRING
```

LISTING 3.

```
/* This function displays: */
/* The positions are: 4 29 51 82 */

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

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 = strsrch( stg1, stg2 );
    printf( "The positions are:\t" );
    while( -1 != *ptr )
        printf( "%d\t", *ptr++ );
    printf( "\n" );
    return( 0 );
}
```

Q. How many programmers does it take to maintain a MAKE dependency file?

A. NONE! If you use VersiMAKE™

VersiMAKE™ is a full-featured MAKE utility that includes:

- **Dependency Generation**
Derives your system's dependencies, through analysis of its C and MASM source files. Say goodbye to manual maintenance of MAKE dependency files!
- **Wild Card File Name Matching**
Analyzes an entire collection of source files with a single statement.
- **Nested Include Files**
Handles standard C and MASM "include" conventions, and the INCLUDE environment variable.
- **Powerful Macro Facilities**
Built-in macros, user-defined macros, and environment variables.
- **Analytical Reports**
Shows the entire include file hierarchy for each source file analyzed, and all of the parent source files for each include file.

Q. How many programmers does it take to trace a symbol thru your system?

A. ONE! If you use VersiCREF™

VersiCREF™ is a unique utility that creates a Master Cross-Reference of your entire system.

- **Multi-Lingual**
Handles C, assembler, or both.
- **Flexible**
File names with line numbers, or file names alone. Global and local symbols, or globals alone.
- **Powerful**
Easily handles systems containing 100 source files or more.



VersiMAKE™ \$125
VersiCREF™ \$75
Both \$150
(Outside U.S., add \$5 for shipping and handling)

800-334-4096
(In NJ, 609-871-0202)
MC/VISA/AMEX accepted

SUMMIT INFORMATION SYSTEMS, INC.
73 East Lane, Willingboro, NJ 08046

Circle No. 34 on Inquiry Card