Mecánica Computacional Volúmen XIII
Compilado por M. Vénere
Bariloche, setiembre de 1992

SparsEx: OOP para el manejo de Sistemas Lineales en Memoria Extendida

Ing. Juan Luis Almará

Prof. Tit. Computación I. Bío-Ingeniería. UNER.

Prof. Tit. Informática II. Ing. Electrónica. UTN Reg. Paraná.

San Lorenzo 41. Depto 2. 3100. Paraná. Entre Ríos, Argentina. Tel. (043) 22 5903.

RESUMEN

Se describe en este trabajo el almacenamiento y acceso a elementos de sistemas lineales esparsos, residentes en memoria extendida de equipos basados en el microprocesador 80386 y compatibles, bajo DOS.
El conjunto de rutinas, implementado con tecnología OOP en Turbo Pascal 6.0, está basado en el uso de dos objetos: uno de ellos, permite el acceso a la memoria extendida; el otro, gestiona el manejo de sistemas lineales mediante direccionamiento hash.
Han sido codificados, además, otros algoritmos para la solución de sistemas lineales por el método de Crout, y la entrada/salida de los datos.
La precisión de los valores numéricos es definible por el usuario. Con datos reales (11 dígitos decimales), en 1 Mbyte de memoria extendida pueden almacenarse unos 76000 números, siendo el rango de los índices, bidimensional, de 1 a 65535.

ABSTRACT

This paper comments the storage and retrival of elements of sparse linear systems, in extended memory of computers based on the 80386 microprocessor, under DOS.
The set of rutines, implemented in Turbo Pascal 6.0, is based on two objects: one of them accesses the extended memory; the other one manages sparse linear systems, through hash addressing.
Other algorithms have been implemented, such as the solution of linear systems using the Crout method, and the input and output of data.
The precision of the numerical values involved is user definable. With real data types, (11 decimal digits), 1 Mbyte of extended memory can hold about 76000 numbers, in a bidimensional matrix with indexes ranging from 1 to 65535.

INTRODUCCIÓN

En diversas aplicaciones, es frecuente la necesidad de almacenar y recuperar valores de matrices casi vacías, de grandes dimensiones.
Hay varias formas de lograr este objetivo:

- Uso de variables estáticas.
- Acceso a través de variables dinámicas y punteros.
- Mediante una disposición especial de los datos, que facilite su localización.
- Utilizando memoria virtual, esto es, discos magnéticos.

Sin embargo, surgen diversos inconvenientes en el uso de las técnicas mencionadas, algunos de los cuales son:

- Limitaciones de tamaño.
- Complejidad de programación.
- Dificultad de depuración.
- Inflexibilidad.
- Obscurecimiento de los algoritmos fundamentales por la metodología de almacenamiento y acceso.
- Baja velocidad de recuperación y actualización.

El uso de la memoria extendida, frecuentemente desaprovechada en los equipos, debido a las implicancias de su acceso, soluciona una parte de los invenientes mencionados.

Por otra parte, la implementación del manejo de la memoria extendida a través de objetos ayuda a la transparencia, a la mejor comprensión del problema real en vez de su representación, a la reutilización en otras aplicaciones, a un mejor diseño conceptual y una disminución en el trabajo de depuración y puesta a punto.

Mediante la facilidad de Turbo Pascal de generar Units, estas ventajas se ven potenciadas, ya que, aparte de las características de encapsulamiento de la OOP, se agrega el agrupamiento de rutinas dentro de unidades funcionales.

Existen en el mercado varias versiones de utilitarios llamados extensores del DOS, que permiten compilar y ejecutar programas que acceden a toda la memoria disponible; sin embargo, pueden presentar algunos inconvenientes:

- Si bien el uso de la memoria es sencillo, otras operaciones, como el servicio a ciertas interrupciones, pueden verse afectadas.
- Algunos programas requieren de un compilador de línea de comandos, lo que lleva a que la depuración de los programas sea un tanto más dificultosa que en un ambiente interactivo.
- Prestaciones no documentadas, aunque de amplio uso, pueden no dar los resultados esperados.
- El uso de otros sistemas operativos debería ser analizado cuidadosamente, en busca de problemas potenciales.

MEMORIA EXPANDIDA Y EXTENDIDA

La memoria, en las computadoras personales, podría clasificarse de la siguiente manera [1]:

Base:	aquella que se direcciona desde el byte 0 hasta los 640 kB.
Superior:	se encuentra entre los 640 kB y 1024 kB.
Alta:	los 64 kB que se hallan entre 1024 kB y 1088 kB.
Extendida:	arriba de 1088 kB, hasta el límite instalado.
Expandida:	aquella que se puede hacer coincidir, por bloques, con una cierta zona de la memoria base, superior o alta.

En lo que se refiere al almacenamiento de grandes matrices, la memoria base es, frecuentemente, muy pequeña; la memoria superior se halla ocupada para diversas tareas: memoria de video, plaquetas emuladoras, bloques de memoria expandida; la memoria alta tiene sólo 64 kB; la memoria expandida es una opción viable, aunque debe considerarse la sobrecarga originada por los algoritmos de cambio de página.

Por estos motivos, es interesante el uso de la memoria extendida: su límite es sólo la capacidad instalada, que, cada vez, tiene un costo menor; su acceso puede realizarse mediante rutinas que trabajen en modo real; existen normas para su uso, de modo que varias aplicaciones podrían compartirla.

ACCESO A LA MEMORIA EXTENDIDA

El acceso a la memoria extendida implica, básicamente, el uso de las siguientes cinco operaciones:

- Determinación de la memoria disponible, y total.
- Obtención de cierta cantidad de memoria para un uso específico.
- Grabación de datos.
- Recuperación de valores.
- Liberación de bloques de memoria.

Existen por lo menos dos metodologías de acceso a la memoria extendida. Ambas hacen uso de facilidades documentadas, y se ajustan a cierto entorno de trabajo:

a. Interrupción 15h: El sistema DOS/BIOS de las computadoras personales utiliza diversas interrupciones de hardware y de software para su funcionamiento. El uso de las mismas se basa en especificar ciertos parámetros en los registros del microprocesador, y luego hacer la llamada adecuada. La interrupción 15h es la que permite acceder a la memoria extendida, y tiene facilidades para lograr diversos objetivos [2].
b. Norma XMS: Las especificaciones de memoria extendida (XMS) son una alternativa, más moderna, para lograr la administración de la memoria sobre 1 Mb [3]. La norma es de uso público, y está incorporada en varios programas comerciales.

Se pueden encontrar otras formas de acceso, que utilizan facilidades no documentadas del sistema operativo o de los microprocesadores. Dependiendo de las aplicaciones, puede ser o no conveniente su uso, quedando siempre latente el riesgo de que la facilidad sea retirada en futuras versiones.

DIRECCIONAMIENTO HASH

La técnica de búsqueda de transformación de clave a dirección (hashing) consiste en usar la identificación de un elemento, tal como su fila y columna, para encontrar, en una tabla lineal, su contenido. La tabla donde se busca puede contener, generalmente, más elementos de los que se van a almacenar, y su dimensión es un número primo. Mediante una función de desmenuzamiento, tal como la división modular, transforma la identificación en un número de elemento de la tabla, lo que posibilita un rápido acceso:

identificación ----> función hash ---> nro. elemento ----> valor

Se han ensayado varias alternativas para lograr un número pseudo-aleatorio que dependa de la identificación, y esté comprendido dentro de la dimensión de la tabla hash [4][5]. Algunos de ellos son:

- División modular.
- Centro del cuadrado.
- Plegado y suma de partes.

Por buena que sea la función de transformación de clave a dirección, usualmente se producen sinónimos o colisiones. Para resolver esta situación, existen dos técnicas:

- Direccionamiento abierto: al producirse una colisión, se busca en otras posiciones de la misma tabla.
- Encadenamiento: el número resultante de la función hash es el puntero a una lista de sinónimos.

Para el caso de direccionamiento abierto, hay varias posibilidades al momento de determinar la posición adecuada: prueba lineal, donde se busca secuencialmente, el primer lugar libre; prueba cuadrática, en que se busca salteando cuadráticamente módulo tamaño de la tabla; y doble direccionamiento hash, donde la próxima búsqueda se realiza mediante una segunda función hash.

El direccionamiento hash permite recuperar con rapidez un valor, pero puede llevar a varios problemas:

- Agrupamientos: se producen cuando las colisiones no se solucionan de una manera efectiva.
- Pérdida de eficiencia: debido a que se deba aplicar varias veces la técnica de búsqueda, por una elección incorrecta de la función hash. Se puede demostrar que el número de pruebas depende del factor de carga [5].
- Borrado: la implementación de los algoritmos se complica si debe considerarse la posibilidad de eliminar elementos.
- Límites: al ser fijo el tamaño de la tabla, e ir incrementándose la cantidad de elementos ocupados (factor de carga), el esfuerzo de búsqueda es mayor. Por otra parte, es difícil de ampliar, ya que hay que realmacenar todos los elementos.
- Enumerabilidad: los elementos se ubican en posiciones que no estan relacionadas con un orden en el valor de las claves, por lo que la enumeración según claves ascendentes, por ejemplo, no es sencillo de lograr.

Sin embargo, a pesar de estos inconvenientes, si se elige adecuadamente la dimensión, de modo que se llene en un 80%, se seleccionan funciones hash que produzcan una buena dispersión de valores, y se programan las posibilidades del borrado de elementos, esta metodología ofrece la ventaja de ser compacta y muy rápida.

SOLUCIÓN DE SISTEMAS LINEALES

El método de Crout es una modificación de la reducción de Gauss, y presenta la ventaja de que se minimiza el almacenamiento de resultados auxiliares. Cada valor, en la matriz de trabajo, se obtiene a partir de elementos previamente calculados, a través de una secuencia contínua de operaciones, que no necesitan datos intermedios [6].

Para facilitar la operatoria, el vector de términos independientes se almacena como la columna n+1 del sistema n x n.

La estructura general del proceso es: los n elementos de la primera columna se determinan primero; luego, los n valores de la primera fila. A continuación, los restantes n-1 elementos de la segunda columna y de la segunda fila, luego los restantes n-2 de la tercera fila y columna, hasta la fila y columna n. Las transformaciones se realizan con los valores de la matriz y los términos independientes.

Este esquema podría resumirse en la siguientes ecuaciones. Las variables marcadas con un tilde son de la etapa previa:

$$a'_{ij} = a_{ij} - \sum_{k=1}^{j-1} a'_{ik}a'_{kj} \qquad (i \geq j)$$

$$a'_{ij} = \frac{1}{a'_{ii}}\left[a_{ij} - \sum_{k=1}^{i-1} a'_{ik}a'_{kj}\right] \qquad (i < j)$$

$$c'_i = \frac{1}{a'_{ii}}\left[c_i - \sum_{k=1}^{i-1} a'_{ik}c'_k\right]$$

$$x_i = c'_i - \sum_{k=i+1}^{n} a'_{ik}x_k$$

Las incógnitas se calculan desde la fila n hacia arriba, con sustituciones.

Se puede demostrar que los elementos a la derecha de la diagonal principal, en la matriz resultante, son idénticos a los que aparecen en posiciones equivalentes en la reducción de Gauss. Los espacios que ocuparían unos y ceros, son ahora usados para datos auxiliares.

Por otra parte, se puede verificar que la matriz resultante se podría descomponer en una triangular inferior, y en una triangular superior, con unos en su diagonal principal.

También, el determinante del sistema de ecuaciones es el producto de los elementos de la diagonal principal de la matriz resultante, por lo que este método es, además, una eficiente manera de evaluar determinantes.

IMPLEMENTACIÓN

Los algoritmos fueron implementados en una computadora personal basada en el microprocesador 80386, 40 Mhz, modo turbo, con 4 Mb de memoria instalados, codificados en Turbo Pascal 5.5 y 6.0 [7] [8].

En base al anális del problema, los procesos se agruparon del siguiente modo:

1. Una unit contiene el objeto básico, que sólo transfiere bytes entre la memoria base y la memoria extendida, y viceversa. Su constructor inicializa los punteros adecuados, y los valores de la tabla de descriptores globales. Debido a que no hay otras referencias, este objeto es reutilizable, para hacer movimientos de cierto número de bytes, independientemente de su significado. Sus métodos básicos leen y graban datos en la memoria extendida.

2. Otra unit contiene un descendiente de este objeto, personalizado de modo que permita intercambiar un registro. El mismo se compone de las coordenadas del elemento de la matriz, fila y columna, su valor, y un caracter indicador del estado: disponible, ocupado, borrado. Este objeto tiene varios métodos, aparte de la lectura, grabación y actualización de un valor: solución de un sistema de ecuaciones, lectura y grabación de todos los elementos en archivos de texto, inicialización a cero de la matriz, generación de sistemas al azar con solución conocida, vuelco de memoria.
3. El programa principal, que usa las units mencionadas, contiene la declaración e inicialización de un objeto lineal, y su uso para diversos fines: pruebas, verificación de resultados, medición de tiempos.

El acceso a la memoria extendida fue implementado a través del uso de la interrupción $15. Como elementos de interés, deben comentarse que el código de acceso debe ser $93, datos, y que la cantidad de bytes a intercambiar debe ser par. La tabla de descriptores globales se llena según se indica en [2], teniendo en cuenta el orden de los bytes que representan una dirección de memoria.

El direccionamiento hash se hace en base al valor de la fila y columna de cada elemento de la matriz, mediante multiplicación y operación módulo tamaño de la memoria asignada. Las colisiones se resuelven mediante una segunda función de hashing, distinta a la anterior. Fue adicionada una rutina para calcular números primos, con el objeto de mejorar el tiempo de respuesta. La inserción de valores se hace comprobando la dirección indicada por la primer función, verificando si el lugar está disponible o borrado, y utilizando la segunda función hasta encontrar un lugar, o determinar, al cabo de cierta cantidad fija de intentos, que la tabla está llena. Esto puede significar que las rutinas no accedieron a espacio disponible después de intentarlo, aunque el mismo, realmente, se encuentre allí.

La rutina de generación de matrices de ensayo, al azar, no presenta alternativas distintivas, salvo la posibilidad de crear arreglos con sólo una banda de valores en la diagonal principal. Los procedimientos de grabación y lectura en archivo de texto son comunes.

Fueron programadas tres versiones del algoritmo de Crout: completo, con pivoteo parcial; completo, sin pivoteo, y solución de sistema con banda de valores. El pivoteo implica la transposición de filas, lo que lleva a que la banda de valores en la diagonal principal se vea distorsionada, lo que implica un mayor tiempo de proceso. El vector de los términos independientes se encuentra en la columna n+1, y la solución del sistema lineal en la columna n+2. Esto facilita el tratamiento uniforme de los datos.

Otras rutinas de interés son la puesta a cero de todos los elementos, el vuelco de memoria, la búsqueda de un valor basado en su fila y columna, la actualización del mismo una vez encontrado, y la determinación del tiempo transcurrido, con fines de comparación.

RESULTADOS

La implementación comentada podría ser analizada desde los siguientes puntos de vista:

a. Ingeniería de software: han sido utilizadas las posibilidades que ofrece el Turbo Pascal: creación de objetos, descendientes, herencia, encapsulamiento, units, y el uso de rutinas del sistema operativo. Las direcciones de memorias intermedias y tablas para la transferencia de datos con la memoria extendida se indican con punteros. La estructura de registros para el uso de interrupciones y otros datos relevantes forman parte del objeto; sin embargo, algunas variables se han declarado como globales.

b. Direccionamiento: el acceso por técnicas hash, usando números aleatorios como datos de trabajo, muestra que, para un llenado del 80%, se requieren de 2 a 3 pruebas para encontrar un determinado valor. El tiempo de acceso, para una prueba, es de unos 0.15 milisegundos. Debido a que se realizan varios intentos para localizar un valor, este tiempo se incrementa en la práctica.

c. Generación de datos: el programa fue probado mediante la generación de sistemas de ecuaciones lineales al azar, con números de precisión real (6 bytes), con solución unitaria para todas las incógnitas. Esta rutina permite seleccionar el ancho de banda del sistema a generar, manteniendo conocida la solución.

d. Precisión: trabajando con representación numérica de 11/12 dígitos decimales, en un sistema de 1024x1024, ancho de banda 15, se obtienen, aún, 7 cifras significativas. La solución se obtiene, simplemente, accediendo a la columna n+2 de cada fila. Como subproducto, el método de Crout permite calcular, también, el determinante, y el programa lo realiza basado en un parámtro lógico. La precisión del determinante es extended (19/20 dígitos decimales, grandes exponentes), y es opcional, porque para ciertos sistemas, se podría producir overflow numérico.

e. Velocidad: depende del número de ecuaciones, del ancho de banda, de las funciones de hashing, del grado de llenado de la memoria asignada, de otro software para el manejo de memoria que haya instalado, de la precisión requerida. A través de mediciones, y para sistemas de 512x512 a 1024x1024, con anchos de banda de 3 a 21, sin otro software instalado, llenando un 30% de la memoria asignada, con números reales (6 bytes), el tiempo de solución, en segundos, podría aproximarse con la fórmula

$$t \sim n\,(a+2)^2 / K$$

donde n es el número de ecuaciones, a es semi-ancho de banda, y K, para las condiciones y máquina indicadas, tiene un valor de, aproximadamente, 500. Esto significa que un sistema de 1024x1024, ancho de banda 13, requiere unos 130 segundos para ser solucionado.

f. Limitaciones: algunas incompatibilidades se presentaron al probar el programa en equipos basados en el 80286, y con la presencia de otros manejadores de memoria como HIMEM.SYS y EMM386.EXE. La cantidad de valores a almacenar depende de la memoria instalada, y es, aproximadamente, de 76000 por cada megabyte. La velocidad se ve reducida cuando se incrementa el factor de llenado.

CONCLUSIONES

Probablemente, no hay (aun) una metodología de almacenamiento de grandes cantidades de datos en memoria RAM, en computadoras personales, que, a la vez, sea:

- Exenta de incompatibilidades.
- Ajustada a un standard.
- Aplicable a los diversos microprocesadores de la familia 80x86.
- Eficiente.
- Aceptada.
- Documentada.
- Utilizable fácilmente desde lenguajes difundidos de alto nivel.

El presente aporte es un posible acercamiento, basado en prestaciones documentadas y OOP. El acceso por interrupciones y el direccionamiento hash llevan a que su tiempo de respuesta no sea el mejor, situación que se revierte, en parte, por las mejoras introducidas en la implementación del método de Crout para matrices esparsas, formadas por bandas alrededor de la diagonal principal.

El manejo de sistemas lineales esparsos en memoria extendida implica la posibilidad de usar la metodología aquí comentada en otras aplicaciones, aparte de sistemas de ecuaciones:

- Bases de datos gráficas.
- Método simplex.
- Grafos y redes.
- Procesamiento de imágenes.
- Sistemas eléctricos de potencia.
- Matrices de costos.

REFERENCIAS

[1]. Forney, J.: "MS-DOS beyond 640 K". McGraw-Hill, 1989.

[2]. Sargent, M. y Schoemaker, R.: "The IBM PC from the inside out". Addison-Wesley. 1986.

[3]. Microsoft Corp., et al.: Extended Memory Specification (XMS), Versión 2.0. Box 97017, Redmond, WA 98073, (206) 882-8080.

[4]. Tenenbaum, A. y Augenstein, M.: "Estructura de Datos en Pascal". Ed. Prentice-Hall Hispanoamericana S.A. 1983.

[5]. Lipschutz, S.: "Estructura de Datos". Serie Schaum. Ed. McGraw-Hill. 1987.

[6]. Hildebrand, F. B.: "Introduction to Numerical Analysis". Tata McGraw-Hill, 1974.

[7]. Borland Intl.: Manual de Refencia Turbo Pascal 6.0.

[8]. O'Brien, S.: "Turbo Pascal 6.0. Manual de Referencia". Ed. Osborne/McGraw-Hill. 1991.