DISEÑO Y APLICACIÓN PRÁCTICA DE UNA ESTRUCTURA FLEXIBLE UTILIZANDO PROGRAMACIÓN ORIENTADA A OBJETOS.

Paul Lorena L.*

RESUMEN

El presente articulo describe el desarrollo e implementación del software de una nueva estructura flexible (Multianillos), que maneja información y datos abstractos. Se utiliza el paradigma de la Programación Orientada a Objetos (POO), para su aplicación práctica en un programa escrito en C++ para un dispositivo de telefonía celular. Esta misma estructura podrá ser utilizada para otros dispositivos móviles como Handhelds inalámbricos, etc, que requieran el uso de nuevos algoritmos para un uso eficaz de la memoria. Se utiliza el lenguaje C/C++, en un compilador Borland C++ 4.5.

Palabras clave : Multianillos. Programación. Estructura flexible.

ABSTRACT

This article describes software development and implementation for a new flexible structure (Multi rings) that handles abstract data and information. The Object Oriented Programming (OOP) Paradigm is used for its practical application in a C++ language program for a cellular telephone device. This same structure could be employed by other mobile devices such as wireless handhelds that require the use of new algorithms for an efficient memory use. The C++ language in a Borland C++ 4.5 compiler is used.

Key words : Multi rings. Programming.

INTRODUCCIÓN

En la actualidad la distancia entre las computadoras personales y los dispositivos móviles se esta acortando, las innovaciones tecnológicas en el campo del hardware, en el área del software la implementación de la Programación Orientada a Objetos (POO) y el desarrollo de nuevos algoritmos permite que se disponga de computadoras, y dispositivos portátiles mas potentes.

Las demandas actuales de información rápida en reducidos espacios de memoria, son un reto continuo para la Ingeniería del Software, de allí surge la necesidad de crear algoritmos mas eficientes en el uso de los recursos de la computadora.

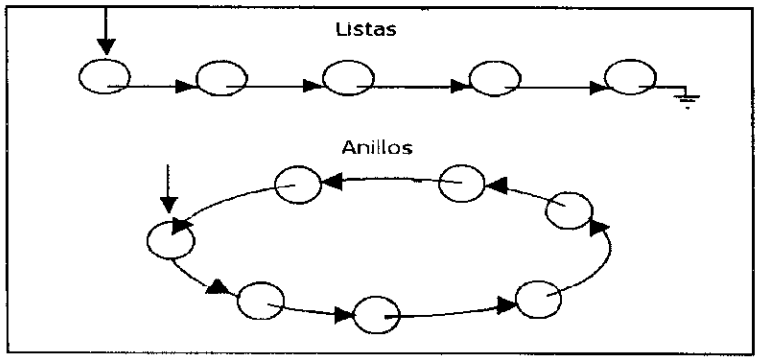


Figura 1. Estructuras Flexibles Tradicionales

*Instituto de Investigación de la Facultad de Ingeniería Industrial. UNMSM.
E-mail . paul_lorena@yahoo.co

ESTRUCTURAS FLEXIBLES

Se denomina así al conjunto de nodos o paquetes de información que están unidas unas a otras por medio de punteros de memoria, éstas pueden ser: las pilas, colas, pilas, y anillos, que se diferencian entre sí por la forma de edición, recuperación y eliminación (gestión) de los nodos.

Cada nodo o paquete contiene información, y que es almacenada en tiempo real en la memoria RAM de la computadora, puede ser posteriormente almacenada en otra unidad física o disco.

PROGRAMACIÓN ORIENTADA A OBJETOS

Según este paradigma de programación, el desarrollo del software se basa en la concepción del objeto, es decir un ente que posee propiedades y métodos. La Programación Orientada a Objetos, se considera una evolución natural de los lenguajes de computación, y su implementación en el mundo de la Ingeniería del Software equivale a la aparición de la máquina del vapor. La POO resuelve con asombrosa facilidad problemas complejos, que no podían ser solucionados eficientemente con la programación estructurada. La POO posee 4 características fundamentales: Abstracción, Herencia, Polimorfismo, y Encapsulamiento.

Los algoritmos necesarios para construir una estructura flexible pueden ser escritos con programación estructurada o secuencial. El proceso de desarrollo y análisis se hace más fácil si se utiliza la programación orientada a objetos.

Desarrollo de una Nueva Estructura Flexible (Multi Anillos).

Se observa que las estructuras tradicionales no son completamente flexibles, la unidireccionalidad es una de sus principales desventajas (ver Figura 1).

El desarrollo de una nueva estructura deberá permitir que el flujo de información o de exploración sea mutidireccional.

La estructura Multi Anillos posee un diseño de Nodo mejorado.

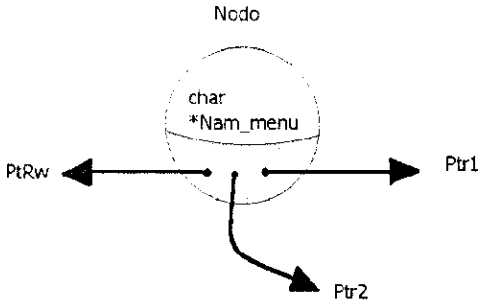


Figura 2. Nueva Estructura Nodular

Esto lo logramos modificando el diseño de la estructura del Nodo (en C++).

```
struct Browser_Node
{
char Name_Menu[25];
Browser_Node * Ptr1;
Browser_Node * Ptr2;
Browser_Node * PtRw;
};
```

Un puntero principal (Ptr1, que apunta al Nodo siguiente), un puntero al nodo anterior (PtrW), un puntero a una estructura lateral (Sub Anillo) (Ptr2). Como se ve estos punteros son recursivos, porque son punteros dinámicos de memoria de su propia estructura.

Como se ve en la figura 3, la estructura Multi Anillos posee una Estructura Principal y unida a ella pueden ir mas estructuras de su mismo tipo (Sub Anillos), que se encuentra unidas dinámicamente por medio de apuntadores de memoria.

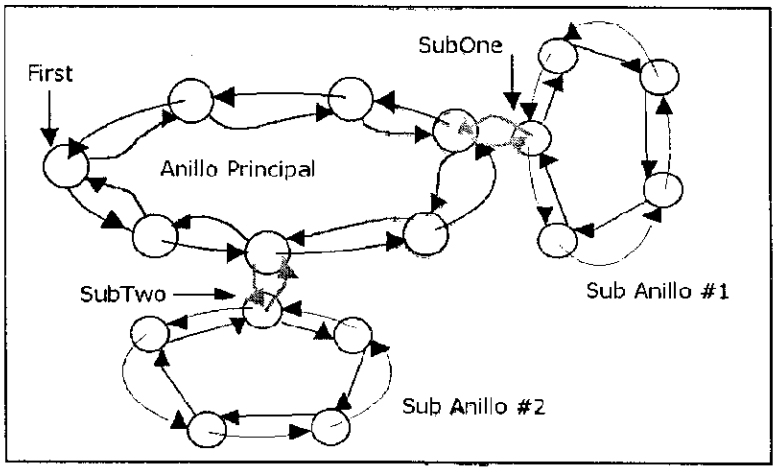


Figura 3 Estructura Multi Anillos

Tambien se puede observar que los Multi Anillos son multidireccionales al poseer punteros en ambos sentidos, lo que le da mayor flexibilidad tanto al desarrrollador como al usuario final del programa.

El algoritmo en C++, que permite la creación de la estructura principales:

```
void CellBrowser::InsertNodo (char *nam, int
nnode)
{
        BrPtr Temp;
        Temp=new Browser_Node;
        strcpy(Temp->Name_Menu ,nam);
        Temp->num =nnode;
        if (First==NULL)
        {
                First=Temp;
                Cursor=Temp;
        }
        else
        {
                Cursor->Ptr1=Temp;
                Temp->PtRw =Cursor;
                Temp->Ptr1 =First;
                First->PtRw=Temp;
                Cursor=Temp;
        }

}
```

El algoritmo en C++, que permite la creacion de los Sub Anillos, es:

```
void CellBrowser::InsertSubNodo (char *nam, char
*nodo,int nnode)

{
        BrPtr Temp, Aux=Cursor;
        Temp=new Browser_Node;
        strcpy(Temp->Name_Menu,nam);
        while (strcmp(Aux->Name_Menu,nodo)!=0)
  {
                Aux=Aux->Ptr1 ;
  }
        if ( SubOne==NULL && nnode==1 )
  {
                Aux->Ptr2 =Temp;
                SubOne=Temp;
                SubOne->PtRw=Temp;
                Temp->Ptr2=Aux;
        }
        else if(SubOne!=NULL && nnode==1)
  {
                (SubOne->PtRw)->Ptr1=Temp;
                Temp->PtRw=SubOne->PtRw;
                Temp->Ptr1 =SubOne;
                SubOne->PtRw=Temp;
        }
}
```

APLICACIÓN PRÁCTICA: APLICACIÓN DE MULTI ANILLOS A UN DISPOSITIVO DE TELEFONIA CELULAR.

El presente trabajo contiene la aplicación práctica de los Multi Anillos, que es utilizado para:

- Gestión del Menú y Sub Menús de un teléfono celular.
- Gestión de la Agenda de Datos del teléfono celular.

Para la gestión de los Menús y Sub Menús del celular se utilizara Multi Anillos con una estructura principal, y sub anillos (Sub Menús).

Para la gestión de la agenda de teléfonos y nombres del usuario, se utiliza Multi Anillos para la Lista Principal. Se desarrolla el programa en C/C++, y con POO.

CARACTERÍSTICAS DEL TELÉFONO CELULAR

- El teléfono celular posee un display (pantalla LCD) de 3 líneas.
- Posee 3 botones principales (Yes, Cancel y No), y 2 botones de exploración (↑, ↓).

El Menú principal deberá tener las siguiente opciones y sub-opciones:

Cuadro 1. Requerimientos del Menú

Opción Principal	Sub Opción
AGEND	
STORE	
RECALL	
CONFIGURATION	ALARM TIME YEAR DAY MONTH SLEEP
BLOQUEOS	CALL OUTS CALL IN CALL HOME ALL CALLS
SEND EMAIL	
READ EMAIL	
DELETE	
CALL	

- Exploración del Menú y Sub Menú se realiza con las teclas de exploración para poder acceder a cualquiera de las Opciones principales se presiona (YES), para salir se presiona (NO), y para salir de la exploración completamente se presiona (CANCEL).

Los programas estan contenidos en los Archivos Multiple.h, y Multiple.cpp, en el archivo de cabecera Multiple.h, están escritos el diseño de las estructuras de 2 objetos, un objeto se encarga de la gestión de los Menús (**class** CellBrowser), y el segundo se encarga de la gestión de la Lista de teléfonos (**class** Nokia), cada uno de ellos tiene una estructura o Nodo parecido (**struct** Browser_Node, y **struct** Data).

Nota: Estos 2 archivos, asi como el codigo compilado, puede ser descargado libremente de: http://www.unmsm.edu.pe/iifi/download

```
/*Archivo: Multiple.h
Archivo de Cabecera FII UNMSM 2001 Paul Lorena*/
#ifndef Multiple_H
#define Multiple_H
#include <iostream.h>
#include <string.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <iomanip.h>
#include <time.h>
#define TRUE 1
#define FALSE !TRUE
//Estructura del Nodo para el objeto que gestiona
Menú y Sub Menú
struct Browser_Nodo
                {
                  char Name_Menu[25];
                  int num;
                  Browser_Node * Ptr1;
                  Browser_Node * Ptr2;
                  Browser_Node * PtRw;
                 };
typedef struct Browser_Node *BrPtr;

class CellBrowser       //Definición del objeto que
gestiona Menú y Sub Menú
                {
        private:
          BrPtr First;
          BrPtr SubOne;
          BrPtr SubTwo;
          BrPtr Cursor;
          BrPtr Flag;
        public:
          CellBrowser();
          ~CellBrowser();
          void InsertNodo(char *, int );
          void InsertSubNodo(char *, char*,int );
          void Up();
          void Down();
          void Test();
          int  Yes();
          void No();
          void Init();
                };
struct Data             //Estructura del Nodo para el
                        objeto que gestiona la Agenda
 {
 double number;
 char name[25];
 int  pos;
 Data *next:
 Data *prev;
 };
class Nokia             //Definición del objeto que
                          gestiona Agenda
 {
 private:
        Data *Item;
        Data *First;
 public:
        Nokia();
        ~Nokia();
        void Insert(double, char*,int );
        double Recall(char *);
        void Recall_Name(char *);
        void DeletePhone(char *);
        double ListAgend();
        int Number_Node();
        int  ListEmpty();
 } ;
#endif
```

UBICACIÓN DE LOS NODOS EN LA MEMORIA.

La memoria en las computadoras constan de cierto número de celdas (o posiciones), cada una de las cuales puede almacenar una porción de la información. Cada celda tiene un número asociado denominado dirección, esta expresado en notación hexadecimal. Se entiende pues, que una celda de memoria es la menor unidad direccionable.

En este caso los Nodos poseen una dirección de memoria, y está definida por las siguientes líneas de comando:

```
BrPtr Temp;
Temp=new Browser_Node;
```

En la primera linea definimos la variable Temp del tipo definido BrPtr (Nodo), en la segunda línea asignamos a Temp -por medio del comando **new**- un espacio de memoria, capaz de contener al **struct** Browser_Node.

La dirección del primer Nodo es 425F:0802 y la del segundo Nodo es 425F:07D6, las posiciones de memoria son consecutivas, la diferencia entre las dos direcciones nos dará el tamaño del Nodo:

```
425F:0802 - 425F:07D6 = 44 bits
struct Browser_Node
                {
                 char Name_Menu[25];
                // char ocupa 1 bit*25= 25 bits
                 int  num;
                // int   ocupa            16 bits
```

```
Browser_Node * Ptr1;
// Puntero de Memoria 1 bit
Browser_Node * Ptr2;
// Puntero de Memoria 1 bit
Browser_Node * PtRw;
// Puntero de Memoria 1 bit
};
```

De esta manera comprobamos que el tamaño del Nodo es de 44 bits, que corresponde al tamaño asignado por el comando new.

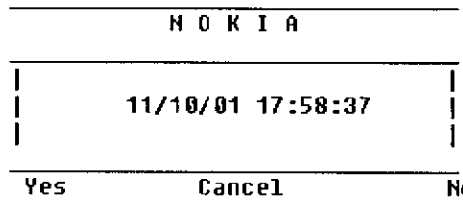
CONSTRUCCIÓN Y DESTRUCCIÓN DE OBJETOS

El algoritmo de los Multi Anillos, permite que los 2 objetos sean creados solo cuando se les necesita, esto es cuando el celular esta en modo Stand By, no existen los objetos en la memoria volátil del dispositivo, lo que posibilita un uso eficiente de la memoria.

El código que permite la construcción del código aparece codificado con el mismo nombre del objeto, y el nombre de la función que destruye al objeto esta precedido con el símbolo de negación ~.

VISTAS DEL PROGRAMA

- Modo en Stand By. no existe ningún objeto.

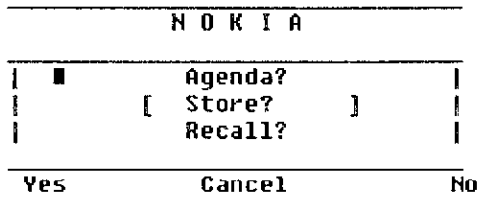


Cuadro 2. Direcciones de Memoria de los Nodos

MENÚ		Dirección de		PUNTEROS	
Opción Principal	**Sub Opción**	**Memoria**	**Ptr1**	**PtRw**	**Ptr2**
AGEND		425F:0802	**425F:07D6**	425F:06A2	NULL
STORE		**425F:07D6**	425F:07AA	425F:0802	NULL
RECALL		425F:07AA	425F :077E	**425F:07D6**	NULL
CONFIGURATION		425F :077E	425F:0676	425F:07AA	425F:0676
	ALARM	425F:0676	425F:064A	425F :077E	425F :077E
	TIME	425F:064A	425F:061E	425F:0676	NULL
	YEAR	425F:061E	425F:05F2	425F:064A	NULL
	DAY	425F:05F2	425F:05C6	425F:061E	NULL
	MONTH	425F:05C6	425F:059A	425F:05F2	NULL
	SLEEP	425F:059A	425F :0752	425F:05C6	NULL
BLOQUEOS		425F :0752	425F:056E	425F:059A	425F:056E
	CALL OUTS	425F:056E	425F:0542	425F :0752	425F :0752
	CALL IN	425F:0542	425F:0516	425F:056E	NULL
	CALL HOME	425F:0516	425F:04EA	425F:0542	NULL
	ALL CALLS	425F:04EA	425F:0726	425F:0516	NULL
SEND EMAIL		425F:0726	425F:06FA	425F:04EA	NULL
READ EMAIL		425F:06FA	425F:06CE	425F:0726	NULL
DELETE		425F:06CE	425F:06A2	425F:06FA	NULL
CALL		425F:06A2	425F:0802	425F:06CE	NULL

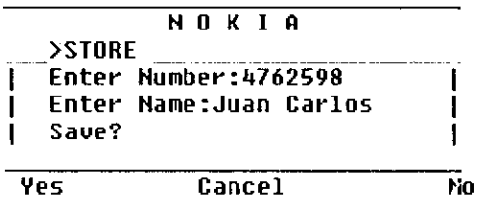
PUNTEROS	DIRECCIONES DE MEMORIA
First	425F:0802
SubOne	425F:0676
SubTwo	425F:056E

- Se presiona ↓ ó ↑, Se creó el objeto CellBrowser.

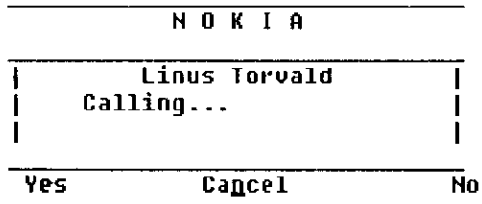


- En cualquier opción se presiona Y [YES], en la Opción Store, se inserta un nodo nuevo a la estructura principal de la Agenda, que fue construida también.

- Cuando se presiona N [NO], se destruye el objeto Agenda.



- Aquí se ve la agenda siendo creada, para luego de ser utilizada ser destruida.



CONCLUSIONES

La tecnología actual, aun no ha podido resolver el problema de la relación tamaño /capacidad en las memorias RAM (memorias volátiles), lo que constituye un obstáculo para los desarrollo de los dispositivos portátiles. El uso de estructuras flexibles, para el software de un dispositivo portátil, es un alternativa viable, a la alta demanda de información y la escasa memoria del sistema.

En la estructura Multi Anillos, los nodos se crean y se destruyen según los requerimientos del sistema.

La multidireccionalidad de los Multi Anillos, posibilitan una rápida busqueda, de un Nodo en particular, constituyendo una ventaja consideralble frente a las estructuras flexibles tradicionales.

Es posible que la estructura principal contenga a otras estructuras como ella, unidas dinámicamente (dada su naturaleza recursiva), es posible que los Sub Anillos posean a su vez otros Sub Anillos que hereden todo el comportamiento de la estructura principal.

Actualmente la empresas tecnológicas que elaboran dispositivos móviles de telefonía celular, han adoptado la Programación Orientada a Objetos, y el estándar Java 2 Micro Edition (J2ME), será utilizado para el desarrollo del software de sus teléfonos celulares, a partir del año 2002.

Este estándar está basado en el Lenguaje Java, desarrollado por Sun Microsistems en 1995, que es un lenguaje de objetos, es decir todo el concepto de POO aplicado directamente a un lenguaje de computación.

BIBLIOGRAFÍA

1. **Deitel H.M. y Deitel P.J., 1995.** Como programar en C/C++ . 2da. ed. Prentice Hall. Mexico

2. **Raffo L., Eduardo. 1998**. Estructuras y Algoritmos en C/C++. Lima.

3. **Ruiz L., Edgar . 1999.** Curso de Lenguaje C. Facultad de Ingenieria Industrial. UNMSM. Lima

4. **Schmuller, Joseph. 1999.** Aprendiendo UML en 24 Horas. Ed. Prentice Hall. México.

5. **Tanenbaum Andrew S. 1985.** Organización De Computadoras Un Enfoque Estructurado, Ed. Prentice Hall. México.