Revista de la Facultad de Ingeniería Industrial
Vol. (8) 1: pp. 66-74 (2005) UNMSM
ISSN: 1560-9146 (impreso) / ISSN: 1810-9993 (electrónico)

UN PROGRAMA EN C++ QUE SIMULA LA CALCULADORA HP 48G

Recepción: Febrero de 2005 / Aceptación: Junio 2005

[1] Paul Lorena Lazo
[2] Edgar Ruiz Lizama

RESUMEN

El artículo presenta un programa en lenguaje C++ que emula el comportamiento de la calculadora HP 48G. Para ello se utiliza los conceptos que provee el paradigma de la programación orientada a objetos y los algoritmos de pila para la notación polaca. En la implementación del programa se ha utilizado el compilador Borland C++ 4.5.

Palabras Clave: Calculadora HP. Notación polaca RPN. Listas enlazadas.

A PROGRAM IN C++ THAT SIMULATES THE HP 48G CALCULATOR

ABSTRACT

The article presents a program in C++ language that emulates the behavior of a HP 48G calculator. For this, we use the concepts that the paradigm of the programming oriented to objects and battery algorithms for the polish notation provide. In the implementation of the program a Borland C++ 4.5 Compiler was used.

Key words: HP calculator. RPN polish notation. Linked list.

(1) Ingeniero Industrial. Miembro del Centro de Investigación y Desarrollo de Software, CIDESOFT. E-mail: paul.lorena@gmail.com
(2) Ingeniero Industrial. Profesor del Departamento de Ingeniería de Sistemas e Informática, UNMSM. E-mail: eruizl@unmsm.edu.pe

INTRODUCCIÓN

Las calculadoras HP y CASIO (usadas mayormente en nuestro medio) trabajan de modo diferente. Las calculadoras CASIO evalúan expresiones en notación INFIJA, por ejemplo si queremos sumar a y b, para obtener un resultado c; ingresamos: a + b; con lo cual obtenemos c. Por el contrario las calculadoras HP evalúan en notación POSFIJA; es decir para sumar a y b ingresamos: a b +; para luego obtener c como resultado. Como se observa en esta última forma de evaluar los operandos se leen primero y el operador va al final; de allí el nombre de evaluación en notación Posfija. Existe una tercera forma de evaluar expresiones que es la PREFIJA; así para sumar a y b se tiene: + a b; es decir primero va el operador y luego los operandos.

Las calculadoras HP son muy útiles para realizar cálculos en ciencias e ingeniería por su alta precisión y porque poseen funciones o rutinas de Biblioteca y hasta su propio lenguaje de programación, permitiendo también el trazado de gráficas de las funciones.

Como una inquietud de los autores del artículo surgió la pregunta ¿Cómo es que trabaja una HP?. Para responder a esta pregunta se escribió el programa cuyo contenido se presenta en su totalidad. El programa utiliza conceptos de Estructuras de Datos y Algoritmos, como son: las listas enlazadas, y los algoritmos de Pila, muy útiles para evaluar en RPN(Reverse Polish Norm). Adicionalmente se hace uso del paradigma orientado a objetos para simplificar la solución.

Por tanto, decidimos crear un programa que haga lo que hace la HP, ese es el aporte a la comunidad informática. A partir de aquí puede irse a cosas más avanzadas.

MARCO CONCEPTUAL

Con el aumento de la demanda de nuevo software, gracias al gran avance tecnológico de los últimos 20 años, los programas de computadora tienden a ser más complejos y a manejar mayor volumen de información. Se llegó al punto en el que los programas que trabajan bajo el concepto de la programación estructurada (C, Pascal, etc.) no podían manejar eficientemente estos datos, muchos nuevos programas no podían crearse por las limitaciones propias de la programación estructurada.

La Programación Orientada al Objeto POO; se concibió para resolver los problemas que la programación estructurada no puede resolver. "El C++ es un lenguaje de programación orientado a objetos..." [1 y 5], si se en-

tiende que C++ es la evolución natural del C, se deduce entonces, que hereda todos los recursos de C.

La POO se sustenta en el concepto que el programador puede crear un nuevo tipo de variable (class), que a diferencia de struct o union en C, es una "entidad lógica"1 conocida como "objeto". Un objeto posee datos privados y funciones miembros, además de poseer un código que los manipula. Dentro del objeto se manipulan los datos privados que no pueden ser modificados desde el exterior, por lo que se protege esta información, el único medio de acceso con el "exterior " – o sea el resto del programa-, son las funciones miembro públicas. Al proceso de manipulación y gestión de datos se le denomina encapsulamiento.

La POO tiene tres recursos importantes: 1. Objetos, 2. Herencia, y 3. Polimorfismo.

Objetos

Son todos las variables definidas por el usuario, que tienen una parte pública y otra privada.

Herencia

La herencia es el proceso por el cual un objeto puede adquirir diferentes propiedades de otro objeto similar. Esto se establece bajo una jerarquía de clases; gracias a ésta herramienta se puede compartir funciones miembros de un objeto o inclusive datos privados de otro similar. La herencia es el mecanismo que hace posible que un objeto sea un ejemplo específico dentro de una clase más general. La herencia permite "la disponibilidad del encapsulamiento" [5].

Polimorfismo

Los lenguajes de programación orientados a objetos son compatibles con el concepto de polimorfismo, que permite usar funciones con el mismo nombre pero con diferente tipo de argumentos. Gracias a esto podemos lograr la: Sobrecarga de funciones, la Sobrecarga de operadores y las Plantillas o Patrones de código.

Se puede declarar dos funciones con el mismo nombre, pero con diferente tipo de argumentos y el programa en tiempo de ejecución podrá depurar y escoger la rutina correcta.

APLICACIÓN PRÁCTICA: PROGRAMA HP_*.CPP

El objetivo del artículo es implementar los algoritmos y el código para emular una calculadora HP. Se presenta el código de los programa Hp_*.cpp, escritos en C++ con el compilador Borland C++ 4.5, de la compañía Borland® que emulan a una calculadora HP con notación polaca (Reverse Polish Norm RPN). Se utilizan listas enlazadas bajo el concepto de la POO, consiguiéndose una versión simple de la calculadora Hewlett Packard 48G.

Originalmente se basó en el programa pila2.cpp [3], que usa lista secuenciales, pero se optó por el uso de lista enlazadas para la implementación dinámica de la memoria.

El programa que se presenta a continuación es Hp_here.cpp; en el la pila y la memoria son diferentes objetos, que comparten los datos privados (protected) y funciones miembros, aplicando el concepto de *herencia* (ver figura 1).

A continuación se muestra el código completo del programa.

```
/*programa que simula a la calculadora HP 48 con
listas enlazadas, usa recurso de HERENCIA.
 hp_plus2 update:  00-03-08/07/08/09/11/12  */
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<string.h>
#include<time.h>
#include<math.h>
#include<dos.h>
#include<stdlib.h>
#include<iomanip.h>
#define FALSE 0
#define TRUE !FALSE
#define ESC '\033'
```

```
struct Nodo {
        double value;
        Nodo *sig;
        char tag[10];
};
//clase base
class Hp {
protected:
        Nodo *primero;  //dato público que sera
compartido
public:
        Hp();        //constructor
        ~Hp();       //destructor
        void push(double);
        double pop();
        void drop();
```

Figura 1a. Listado del programa *hp_here.cpp*

```
        void screen();
        int listaVacia();
};
//clase derivada
class memHp : public Hp {
public:
        void store(double);
        double rcl();
        void del();
        void display();
        void nextview();
};
double valor(double op1, double op2, char c );
double valor(double op2, char c );
double concate(char s1[], char s2[], int MAX);
void lineas_vert(int fila_1,int fila_2, int col, char car);
void lineas_hor(int col_1,int col_2, int filas, char car);
double a_numer2(char s[], int m);
int is_command(char c);
int is_editor(char c);
int is_symbol(char c);
void casehp();
void timer();
void error(char c);
//constructor, destructor y funciones miembros
Hp :: Hp()
{
        primero = NULL;
}
Hp :: ~Hp()
{
        Nodo *p;
        Nodo *temp;
        p = primero;
        while (p!=NULL)
        { temp = p->sig;
          delete p;
          p = temp;
        }
}
void Hp :: push(double carac)
{
        Nodo *p;
        p = new Nodo;
        p -> value = carac;
        p -> sig = primero;
        primero = p;
}
double Hp :: pop()
{
        if(!listaVacia())
        {       Nodo *p = primero;
                primero = p->sig;
                return (p->value);
        }
        else {
                error(' ');
                return 0;
        }
}
void Hp :: drop()
{  if(!listaVacia())
        {  Nodo *p = primero;
           primero = p->sig;
        }
        else
           error(' ');
}
void memHp :: store(double op2)
{ Nodo *p;
        p = new Nodo;
        p -> value = op2;
        gotoxy(18,13);
        cout<<"Mode Store/ ";
        gotoxy(18,14);
        cout<<"Name var?: ";
        cin>>p->tag;
        p -> sig = primero;
        primero = p;
}
double memHp :: rcl()
{       char s[10];
        int encuentra=0;
        gotoxy(18,13);
        cout<<"Mode Rcl/ ";
        gotoxy(18,14);
        cout<<"Name var?: ";
        cin>>s;
        Nodo *p = primero;
        if (listaVacia())
                error(' ');
        else{
                while(!encuentra && p != NULL)
                {
                  if (strcmp(p->tag,s)==0)
                                encuentra = 1;
                        else
                                p = p->sig;
                }
        }
        if (encuentra)
                return p->value;
        else
                return 0;
}
void memHp :: del()
{       int encuentra = FALSE;
        Nodo *p;
        Nodo *antP;
        p = primero;
```

Figura 1b. Listado del programa *hp_here.cpp*

```
        antP = NULL;
        char s[10];
        gotoxy(18,13);
        cout<<"Mode Del/ ";
        gotoxy(18,14);
        cout<<"Name var?: ";
        cin>>s;
        if (listaVacia())
                error(' ');
        else {
             while(!encuentra && p != NULL)
                 {  if(strcmp(p->tag,s)==0)
                          encuentra = TRUE;
                    else {
                          antP = p;
                          p = p -> sig;
                    }
              }
              if (encuentra)
              {   if (antP == NULL)
                    {          primero = p -> sig;
                               delete p;
                    }
                    else {
                               antP -> sig = p -> sig;
                               delete p;
                    }
              }
              else
                    error(' ');
             }
}
// imprime pila
void Hp :: screen()
{   casehp();
            Nodo *p;
            int acu = 0;
            p = primero;
             if (!listaVacia())
             { while (p != NULL)
               {       p = p -> sig;
                       acu++;
               }
               p = primero;
               int fila = 14;
               for(int i=1;i<=acu;i++)
               {       gotoxy(33,fila--);
                       if(fila>=5)
                       { cout.fill(' ');

 cout<<setw(10)<<setprecision(6)<<p->value;
                       }
                else{
                       gotoxy(18,6);
                       cout<<"->• • • ";
                }
               p=p->sig;
               }
   }
}
//imprime flags de memoria
void memHp :: display()
{           Nodo *p;
            p = primero;
            int x=0;
            gotoxy(13,16);
            if(!listaVacia())
            {  while (p != NULL)
                       { if(x<=46)
                                 cout<<p->tag<<"|";
                         x=wherex();
                         p = p -> sig;
                       }
            }
}
//funcion para ver flags de memoria desplazados
void memHp :: nextview()
{           Nodo *p ;
            p = primero;
            int x=0;
            gotoxy(13,16);
            clreol();
            if(!listaVacia())
            {  while (p != NULL)
               { if(x<46)
                        printf("%s|",p->tag);
                       else
                       {  gotoxy(18,13);
                          cout<<"press key";
                          getch();
                          gotoxy(13,16);
                          clreol();
                          printf("%s|",p->tag);
                       }
                       x = wherex();
                       p = p -> sig;
               }
              }
}
//funcion valida para ambos objetos!!!
int Hp :: listaVacia()
{           if (primero == NULL)
                       return TRUE;
            else
                       return FALSE;
}
/*************** function main *************/
void main() // hp_here.cpp
{           char c, t[10],s[15];
```

Figura 1c. Listado del programa *hp_here.cpp*

```
        int k;
        double op1, op2, op3, x, y;
        Hp heap;
        memHp memo;
        casehp();
        c = getch();
        while(c!=ESC)
        {  t[0]=c;
           if(!is_command(c))
           {  gotoxy(18,14);
                 putchar(c);
                 gets(s);
                 k = strlen(s);
                 x = concate(t,s,k);
                heap.push(x);
                 heap.screen();
                 memo.display();
           }
           else {
           if(!is_editor(c))
           { if(!is_symbol(c))              {
op2 = heap.pop();
                        op1 = heap.pop();
                        y = valor(op1, op2, c);
                        heap.push(y);
                        heap.screen();
                        memo.display();
                }
                else{
                        if(c!='m')
                        {       op2 = heap.pop();
                                y = valor(op2, c);
                                heap.push(y);
                                heap.screen();
                                memo.display();
                        }
                        else  ///acceso a memo-
ria...........///
                        { putchar(c);
                          c=getch();
                          switch (c){
                                case 's': op2 =
heap.pop();  // store memory

   memo.store(op2);

   heap.screen();

   memo.display();
                                                break;
                                case 'r': op3 =
memo.rcl();  // recall memory

  heap.push(op3);

   heap.screen();

   memo.display();
                                                break;
                                case 'd':
memo.del(); // delete nodo memory

   heap.screen();

   memo.display();
                                                break;
                                case 'v':  //
nextview

   heap.screen();

   memo.display();

    memo.nextview();
                                                break;
                                default: error(c);
                                }
                          }
                        }
                }
                else {
                switch (c)   //acceso a menu
command_edit
                {       case 'w': {   //swap
                                op1 = heap.pop();
                                op2 = heap.pop();
                                heap.push(op1);
                                heap.push(op2);
                                heap.screen();
                                memo.display();
                                break;
                        }
                        case '\b': {  // drop
                                op3=heap.pop();
                                heap.screen();
                                memo.display();
                                break;
                        }
                        case 'd': {   //delete
                                w   h   i   l   e
(!heap.listaVacia())
                                {       op2   =
heap.pop();
                                }
                                heap.screen();
                                memo.display();
                                break;
                        }
                        case 'e':{    //edit
```

Figura 1d. Listado del programa *hp_here.cpp*

```
                                op2 = heap.pop();
                                heap.screen();
                                memo.display();
                                gotoxy(18,14);
                                cout<<op2;
                                cout<<"  ";
                                break;
                        }
                        case '\r':{     //enter
                                op2 = heap.pop();
                                heap.push(op2);
                                heap.push(op2);
                                heap.screen();
                                memo.display();
                                break;
                        }
                        case 'u':{      //undo
                                heap.push(op3);
                                heap.screen();
                                memo.display();
                                break;
                        }
                }
        }
  }
  gotoxy(18,14);
  c = getch();
  }
}
/**************verifica si el caracter es operador******************/
int is_command(char c)
{       chars[]="+-*/^sd\be!wct\rirCbum";
        int i=0;
        while(s[i]!='\0')
                if(c==s[i++])
                        return TRUE;
        return FALSE;
}
int is_editor(char c)
{       chars[]="wd\be\ru";
        int i=0;
        while(s[i]!='\0')
                if(c==s[i++])
                        return TRUE;
        return FALSE;
}
int is_symbol(char c)
{       char s[]="!sctiCbrm";
        int i=0;
        while(s[i]!='\0')
                if(c==s[i++])
                        return TRUE;
        return FALSE;
}
```

```
/********genera marco de calculadora*********/
void casehp()
{       clrscr();
        int  j=9;
        lineas_vert(3,15,15,'|');
        lineas_vert(3,15,45,'|');
        lineas_hor(14,46,17,'*');
        lineas_hor(15,45,15,'_');
        lineas_hor(15,45,5,'_');
        lineas_hor(15,45,2,'_');
        lineas_hor(15,45,1,'_');
        lineas_hor(14,46,20,'*');
        gotoxy(17,3);
        cout<<"hp 48GX";
        gotoxy(12,18);
        cout<<"Edit[e] Del[d] Swap[w] Drop[<-]
Undo[u]";
        gotoxy(13,19);
        cout<<"[ms_tore]  [mr_cl]  [md_el]
[mv_iew]";
        gotoxy(13,21);
        cout<<" sin[s] cos[c] tan[t] fact[!]";
        gotoxy(13,22);
        cout<<"  1/x[i] x²[r] A_Circ[C]  ±[b]";
        for(int k = 6;k <= 14;k++)
        {  gotoxy(16,k);
          cout<<j<<":";
           j--;
        }
        timer();
        gotoxy(18,14);
}
void lineas_vert(int fila_1,int fila_2 ,int col, char car)
{       for(int rows=fila_1;rows<=fila_2;rows++)
        {       gotoxy(col,rows);
                cout<<car;
        }
}
void lineas_hor(int col_1,int col_2, int filas, char car)
{       for(int col=col_1;col<=col_2;col++)
        {       gotoxy(col,filas);
                cout<<car;
        }
}
/**********genera reloj de calculadora**********/
void timer(void)
{  char datebuf[9];
   char timebuf[9];
   _strdate(datebuf);
   _strtime(timebuf);
   gotoxy(28,4);
  printf("%s %s",datebuf,timebuf);
}
/***funcion para convertir caracteres a números***/
```

Figura 1e. Listado del programa *hp_here.cpp*

```
double concate(char s1[], char s2[], int MAX)
{       int k=MAX, i=0,j=0;
        for(k;k<=MAX*2-1;k++)
        {  s1[++j]=s2[i++]; }
        return(a_numer2(s1,MAX+1));
}
double a_numer2(char s[], int m)
{       int i=0,k=0, f=1;
        char ch;
        double num, acu=0, deci=0 , factor;
        while(i<m && s[i]!='.')
        {       ch=s[i];
                num=ch-'0';
                acu=acu*10+num;
                i++;
                if (s[i]=='-'&& i==(m-1))
                {       f=-f;
                        m--;
                }
        }
        if(s[i++]=='.')
        { while(i<m && s[i]!='-')
          { ch=s[i];
                num=ch-'0';
                deci=deci*10+num;
                i++;
                k++;
                if (s[i]=='.')
                        error('.');
          }
        if(s[i]=='-'&& i==(m-1))
                f=-f;
        factor = pow(10,k);
        deci = deci/(double (factor));
        acu = (acu + deci);
  }
  return (acu*f);
}
/***********biblioteca de funciones**********/
double valor(double op1, double op2, char c )
{ double y;
        switch(c)
        {       case '+': y = (op1+op2); break;
                case '-': y = (op1-op2); break;
                case '*': y = (op1*op2); break;
                case '/': if (op2!=0) y = (op1/op2);
                          else{
                                error(c);
                                y = 0;
                          }     break;
                case '^': y = pow(op1,op2); break;
//...^ ALT[094]
                default : break;
        }
        return y;
}
double valor(double op2, char c )
{       double y=1;
        switch(c)
        {       case '!':
                        int k;
                        for(k=1;k<=op2;k++)
                        {               y*=k;
                        }break;
                case 'c':
                        y = cos(op2*M_PI/180);
break;
                case 't':
                        y = tan(op2*M_PI/180);
break;
                case 's':
                        y = sin(op2*M_PI/180);
break;
                case 'i': if(op2!=0)
                                y = 1/op2;
                          else
                                error('/');
                                break;
                 case 'b':
                          y =-op2;
                          break;
                 case 'r':
                          y = op2*op2;  break;
                 case 'C':
                          y = pow(op2,2)*M_PI;
break;
                 default:  break;
        }
        return y;
}
/***********mensajes de error**********/
void error(char c)
{    gotoxy(18,6);
        cout<<c<<"\a Error:";
        switch (c)
        {  case '/':  gotoxy(18,7);
                        cout<<"Infinitive Result";
break;
   case '.':  gotoxy(18,7);
                        cout<<"Invalid Sintaxis";
break;
             default:   gotoxy(18,7);
                        cout<<"Too  Few
Arguments";
        }
        getch();
}
/******** fin del programa hp_here **********/
```

Figura 1e. Listado del programa *hp_here.cpp*

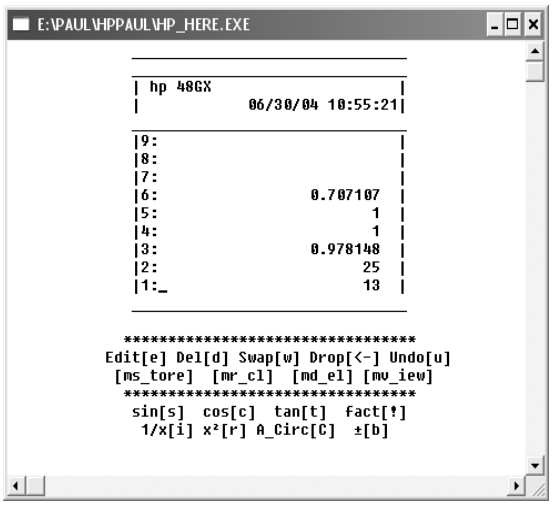

Figura 2 Digitando 45

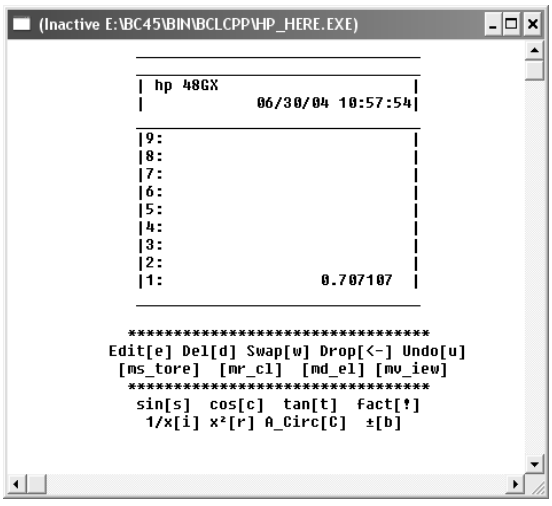

Figura 3. Digitando s para obtener el sen de 45

En las figuras 2 y 3 se presenta la ejecución del programa *hp_here.cpp*.

CONCLUSIONES

Como se mencionó el programa emula a la calculadora HP48G, por lo tanto se ha tratado de reproducir las características más notorias de ésta, estableciendo funciones como:

a. Comandos de edición:

Swap: Invertir el último dato ingresado con el penúltimo de la pila,
Drop: Eliminar el último dato de la pila,
Del: Eliminar todos los datos de la pila, y
Undo: Recuperar el último valor borrado.

b. Comandos de memoria:

Mstore (Memory Store): Almacenar un número en pila creando una etiqueta visible.
Mrcl(Memory Recall): Devolver el valor almacenado previamente.
Mdel(Memory Delete): Borrar de la memoria una variable, se hace el reconocimiento por el nombre de la variable (name var).
Mview(next): Permitir visualizar los flags de memoria desplazados a la derecha del display.

c. Biblioteca de operaciones básicas:

Suma, resta, multiplicación, división, factorización, potencia, seno, coseno, tangente (ingresar ángulos en grados sexagesimales), inversa, cambio de signo(+/-), y cálculo de un área circular.

d. Entrada / Salida

Aquí se ha logrado que la entrada de datos sea lo más parecido posible a la calculadora, comenzando por la visualización de los datos, al estilo H P.
El programa es capaz de distinguir si se ha ingresado un número o un operador ejecutando la rutina adecuada en cada caso.
Se procura generar todos los mensajes de error, igual que la calculadora, y en la versión DOS es posible oír el beep característico.
Finalmente se implementa la fecha y el reloj, así como las etiquetas de los nombres (flags) de las variables guardadas en memoria.

En el programa se ha reunido la mayoría de conceptos de programación: iteración, recursividad, sobrecarga de funciones, tipos abstractos de datos, programación estructurada, y programación orientada a objetos, clases, objetos, polimorfismo, y herencia.

Indudablemente desde el punto de vista de un usuario final para realizar cálculos bastará una calculadora simple y si se posee un sistema operativo como Windows® bastará la calculadora incluida en el, o la herramienta Excel proporcionada por la suite Office, de Microsoft. Sin embargo el aporte del artículo radica en que es posible crear un programa propio que trabaje de la misma manera que lo hacen las calculadoras HP.

Como todo programa, la versión presentada es susceptible de ser mejorada, añadiéndole nuevas funciones y/o mejoras como por ejemplo funciones para graficar expresiones.

BIBLIOGRAFIA

1. Deitel, Harvey. M. y Deitel, Paul. J. (2003). *C++ How to Program.* Fourth Edition. Prentice Hall Inc. U.S.A.

2. Gamma, Erich; Helm, Richard & Others. (1999). *Introduction to Oriented Design in C++*. 1ra. Ed. Addison Wesley Publishing Company, Inc. U.S.A.

3. Raffo Lecca E. (1998). *Algoritmos y estructuras de datos con C/C++*. Raffo Lecca Editores. Lima, Perú.

4. Ruiz L. E. (2004). *Un programa en C++ que implementa grupos abelianos.* En Industrial Data Vol. (7)1: pp. 55-60. Lima, Perú.

5. Stroustrup, Bjarne. (2002). *El Lenguaje de Programación C++*. Edición especial. Adisson Wesley-Pearson Educacion S.A. España.