UNIVERSIDAD NACIONAL EXPERIMENTAL POLITÉCNICA
"ANTONIO JOSÉ DE SUCRE"
VICERECTORADO PUERTO ORDAZ
DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA
SECCIÓN DE COMPUTACIÓN – **COMPUTACION III**

PRACTICA N° 4

OBJETIVOS.

Aprender los conceptos básicos del uso de estructuras de datos dinámicas, listas doblemente enlazadas.

BASE TEÓRICA.

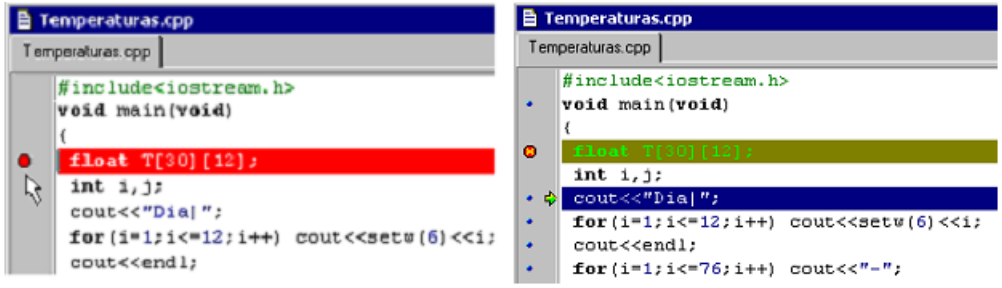
♦ Uso del debugger (depurador).

El termino **debug** en programación se asocia con un proceso de detectar y reducir el numero de errores en un programa, en otras palabras, depurar el programa. El debugger es una facilidad que brinda el lenguaje para realizar este proceso.

Como una introducción al proceso de depuración, se incluyen conceptos básicos que pueden ser aplicados en el desarrollo de la presente práctica. En la medida que se va adquiriendo destreza en el manejo del BCB –IDE y conocimiento del C++, se puede profundizar en el proceso de debugging.

Términos relacionados con el depurador:

Un ***breakpoint*** es una marca en el programa que le indica al depurador que pare la ejecución en ese punto.

Para establecer un breakpoint se coloca el cursor en la banda gris a la izquierda del código y se pulsa el ratón frente a la línea donde se quiere parar.	

El editor pone un punto rojo y resalta la línea con una banda roja. Para quitarlo se vuelve a pulsar sobre el punto rojo y el editor quita el punto y la línea queda normal.

Punto de ejecución es la línea que va a ser ejecutada en el paso siguiente. Cuando se ejecuta paso a paso un programa, el punto de ejecución es resaltado con una banda azul y aparece una flecha verde marcando la próxima línea a ser ejecutada, como se muestra en la figura de la derecha.

Watch (Ver) variables. Para que se detiene la ejecución de un programa.

Usualmente puede interesar observar el comportamiento de las variables. Es importante comprobar que una variable tiene el valor que debe tener o si no se sabe el valor podría ser de interés averiguarlo. Esta función permite esto.

Es posible observar varias variables en una sesión del depurador. Existe una ***Watch List*** donde se ubican las variables que están en observación. Esta lista aparece al usar **Run|Add Watch...** del menú principal. IDE muestra dos ventanas de dialogo. Una con propiedades (**Watch Properties**) y otra con la lista (**Watch List**).

Comandos Relacionados con el depurador:

El comando ***Run->Step Over*** ejecuta un programa línea x línea, o paso a paso, usando tecla F8. La instrucción que se ejecuta es la que está siendo resaltada con la banda azul y el punto de ejecución pasa a la línea siguiente.

Si la instrucción a ejecutar contiene una llamada a una función, se ejecuta la función inmediatamente y el punto de ejecución cambia a la siguiente instrucción. Si se desea ejecutar la función paso a paso se pulsa la tecla F7 al ejecutar la línea que contiene la llamada a la función. En este caso el control de ejecución, banda azul, pasa a la función y con la tecla F8 se procede a ejecutar paso a paso la función. Una vez se llega al final el control, banda azul, regresa a la línea siguiente en el programa que llamó la función.

Si la instrucción a ejecutar es una salida por pantalla (cout) se puede observar el efecto de la instrucción en la consola. Si la instrucción es una entrada del teclado (cin), el programa se detiene a esperar por el dato que va a ser ingresado entonces es necesario ir a la consola, ingresar el dato y una vez que se pulsa la tecla <**enter**> el programa carga el valor en la variable respectiva y continúa la ejecución.

Step Over	F8	Ejecuta el código fuente de la línea (punto de ejecución) y se detiene en la línea siguiente.
Trace Into	F7	Si hay una función en el punto de ejecución, salta al código de la función.
Run to Cursor	F4	Se escoge una línea en el código poniendo el cursor el cualquier punto de la línea, al pulsar F4, el programa se ejecuta hasta la línea escogida.

Funciones para Listas Dobles.-

```
ptNODO ULTIMO(ptNODO Ld)
{     ptNODO A = Ld;
      if(A != NULL)
      {     while (A->Sig!=NULL)  A = A->Sig;
            return  A;        }
      else
      {     cout<<"Lista Vacía";
            return NULL;      }
}
```

```
void MOSTRAR_INV(ptNODO Ld)
{      ptNODO  A;
       A = ULTIMO(Ld);
       while(A != NULL)
       {     cout<< A->Info;
             A = A->Ant;
       }
}
```

```
void INSERTAR_INI(ptNODO &Ld, DATO X)
{     ptNODO NUEVO;
      NUEVO=new(NODO); NUEVO->Info=X;
      NUEVO->Sig=NULL; NUEVO->Ant=NULL;
      if(Ld==NULL) Ld=NUEVO;
      else
      {     NUEVO->Sig=Ld;
            Ld->Ant=NUEVO;
            Ld=NUEVO;
      }
}
```

```
void INSERTAR_FIN(ptNODO &Ld, DATO X)
{     ptNODO NUEVO,U;
      NUEVO=new(NODO); NUEVO->Info=X;
      NUEVO->Sig=NULL; NUEVO->Ant=NULL;
      if(Ld == NULL) Ld=NUEVO;
      else
      {     U=ULTIMO(Ld);
            U->Sig=NUEVO;
            NUEVO->Ant=U;
      }
}
```

```
void MOSTRAR_LISTA (ptNODO Lista)
{      ptNODO A=Lista;
       if (A != NULL)
       {     cout<<"Inicio ->>";
             while(A!= NULL)
             {    cout<<A->Info;
                  A=A->Sig;
             }
             cout<<"<<- FINAL";
             getch();
       }
       else cout<<"Lista Vacia";
}
```

```
void BORRAR_NODO(ptNODO &Ld, ptNODO  &P)
{      if(Ld!=NULL && P != NULL)
       {     if(P==Ld && P->Sig == NULL) Ld=NULL;
             else if(P==Ld)
             {    Ld = Ld->Sig;
                  Ld->Ant = NULL;
             }
             else if(P != ULTIMO(Ld))
             {    P->Ant->Sig = P->Sig;
                  P->Sig->Ant = P->Ant;
             }
             else P->Ant->Sig = NULL;
             delete(P);
       {
}
```

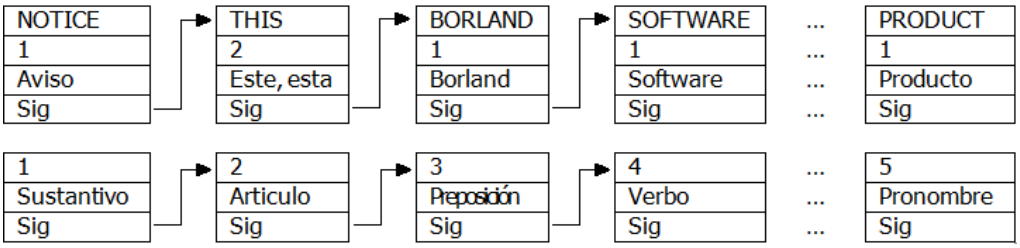
```
void INSERTAR_ORDEN(ptNODO &Ld, DATO X)
{     ptNODO P=NULL, A = Ld,
      NUEVO=new(NODO);  NUEVO ->Info = X;
      NUEVO->Sig = NULL; NUEVO->Ant = NULL;
      if(Ld==NULL) Lista=NUEVO;
      else
      {    while(X >= A->Info && A->Sig != NULL)
           {    P=A;
                A=A->Sig;
           }
           if(X > A->Info) P=A;
           if(P == NULL)
           {    NUEVO->Sig = Ld;
                A->Ant = NUEVO;
                Ld = NUEVO;
           }
           else
           {    if(P->Sig != NULL)
                {    NUEVO->Sig = P->Sig;
                     P->Sig->Ant = NUEVO;
                     P->Sig = NUEVO;
                     NUEVO->Ant = P;
                }
                else
                {    P->Sig=NUEVO;
                     NUEVO->Ant = P;
                }
           }
      }
}
```

```
void INSERTAR_NODO(ptNODO &Ld, DATO X, int POS)
{     int i=1;
      ptNODO A = Ld, NUEVO=new(NODO);
      NUEVO ->Info = X;
      NUEVO->Sig = NULL; NUEVO->Ant = NULL;
      if(POS==1 || Ld==NULL) INSERTAR_INI(Ld,X);
      else
      {     while(i<POS-1 && A->Sig != NULL)
            {     i++;
                  A=A->Sig;
            } //----------------------------->
```

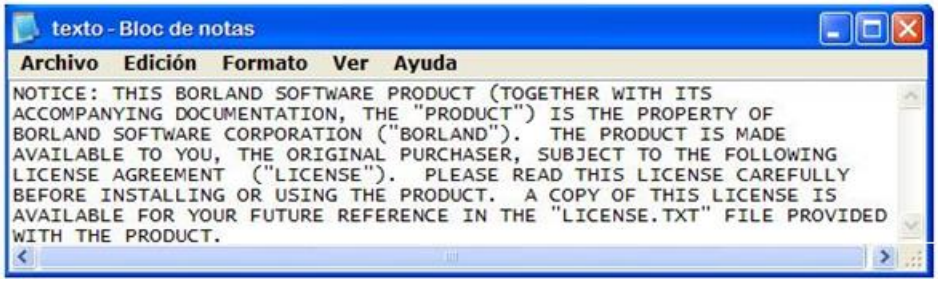
```
//------>  if(A->Sig != NULL)
            {     NUEVO->Sig = A->Sig;
                  A->Sig->Ant = NUEVO;
                  A->Sig = NUEVO;
                  NUEVO->Ant = A;

            }
            else INSERTAR_FIN(Ld,X);
      }
}
```

♦ DESARROLLO DE LA PRÁCTICA.

Escribir un programa que lea un archivo que contiene texto y almacene en una lista doblemente enlazada las palabras que contiene el archivo. Cada palabra es un nodo de la lista junto con otros datos asociados a ella, como en un diccionario, como son: Tipo ([1] sustantivo, [2] articulo, [3] Preposición, etc.) y Significado en español. Los tipos deben ser almacenados en otra lista.



La imagen muestra un archivo de texto que corresponde a un trozo de la licencia de Borland C++. Este archivo se encuentra en el directorio donde está instalado este software.



Si se usa este archivo como los DATOS de la lista se debería observar la siguiente imagen al mostrar la lista:

```
 1          NOTICE
 2            THIS
 3         BORLAND
 4        SOFTWARE
 5         PRODUCT
 6        TOGETHER
 7            WITH
 8             ITS
 9     ACCOMPANYING
10   DOCUMENTATION
11             THE
12         PRODUCT
 ↓
```

El programa que se muestra a la derecha lee un archivo de texto y muestra las palabras que contiene el archivo filtrando solo los caracteres alfabéticos. Puede usar este programa para seleccionar el archivo y cargar los datos.

El programa deberá tener las siguientes opciones:

[1] Cargar un archivo de texto
[2] Mostrar la lista de palabras
[3] Agregar Datos
[4] Buscar palabra
[5] Opción Extra.

```
void main()
{       ifstream A1;
        char aux[25],let;
        int i,flag=0,cont=0;
        A1.open("texto.txt");
        if(!A1)
        {       cout<<"Problemas con archivo";
                getch();
                return;
        }
        A1.get(let);
        while(!A1.eof())
        {       i=0;
                flag=0;
                while(isalpha(let
                {       aux[i]=let;
                        A1.get(let);
                        i++;
                        flag=1;
                }
                if(flag==1)
                {       aux[i]='\0';
                        cont++;
                        cout<<setw(3)<<cont<<setw(20)<<aux<<endl;
                }
                A1.get(let);
        }
        A1.close();
        getch();
}
```