Estructura de Trabajo para el Segundo Examen Parcial Laboratorio de Programación

Prof. Luis Garreta
Pontificia Universidad Javeriana - Cali

1. Fechas

- El exámen se realizará el próximo jueves 4 de abríl y será escrito sobre el tema que viene a continuación.
- El martes 7-9 am se revisarán los proyectos.
- Esta descripción y los recusos se encuentran en el github: `https://github.com/lgarreta/labprog` dentro de examenes

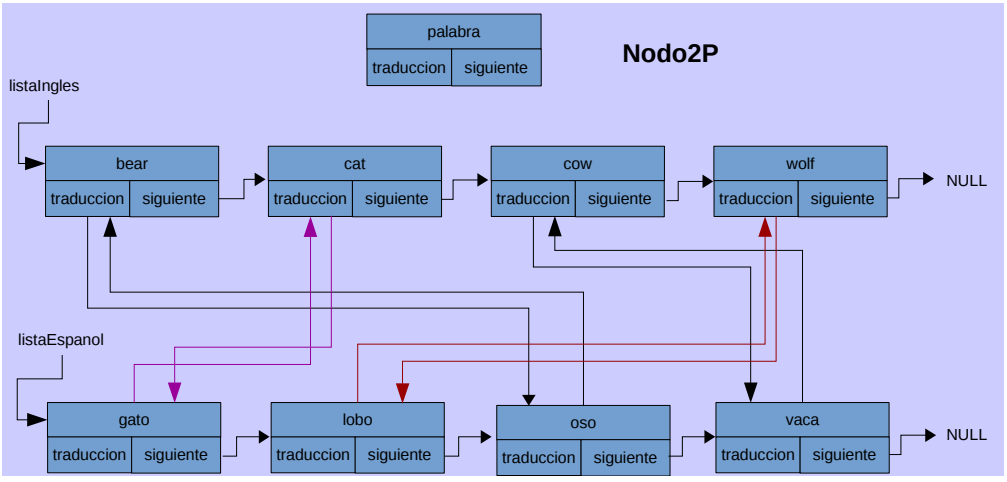
2. Conocimientos a Evaluar:

- Manejo de archivos texto y binarios (lectura/escritura)
- Menejo de estructura
- Manejo de punteros
- Listas enlazadas

3. Descripción

La implementación propuesta como proyecto del diccionario inglés-español con listas enlazadas, carga la lista ordenada lexicográficamente solo por las palabras en inglés y no las de español. Sin embargo, en ciertas aplicaciones se necesita tener ordenada una lista bajo distintos criterios (ej. en una nómina ordenada por salario, por antiguedad, por nombre, etc.), y en nuestro caso queremos que también la lista esté ordenada por las palabras en español.

Para esto proponemos una implementación de dos listas separadas, donde los nodos van a contener ahora solo la palabra, el apuntador al siguiente y un apuntador a su traducción. La palabra va a ser ahora el *info* del nodo, y el apuntador ***traduccion*** va a apuntar al nodo que contiene su respectiva traducción, gráficamente se pueden ver al nodo y a las listas así:



4. Implementación

A continuación se muestra la implementación de las principales estructuras y de las funciones *crearNodo2P, addListas2P* y *cargarDictToListas2P*. Observe que se tienen dos listas globales que son las que van a contener las palabras y por lo tanto las funciones que las modifiquen no van a retornalas ni recibirlas como parámetros.

```c
typedef struct Dict {
  char palIngles  [50];
  char palEspanol [50];
} TipoDict;

typedef struct Nodo2P {
  char palabra [50];
  struct Nodo2P *siguiente;
  struct Nodo2P *traduccion;
} TipoNodo2P;

TipoNodo2P *listaIngles=NULL;
TipoNodo2P *listaEspanol=NULL;

TipoNodo2P *crearNodo2P (char *palabra) {
  TipoNodo2P *nodo = (TipoNodo2P *)malloc (
                       sizeof (TipoNodo2P));
  strcpy (nodo->palabra, palabra);
  nodo->siguiente = NULL;
  nodo->traduccion = NULL;
  return nodo;
}
```

```c
// Inserta de forma ordenada en las dos listas
void addListas2P (char *palIng, char *palEsp) {
  TipoNodo2P *nodoIng = crearNodo2P (palIng);
  TipoNodo2P *nodoEsp = crearNodo2P (palEsp);

  nodoIng->traduccion = nodoEsp;
  nodoEsp->traduccion = nodoIng;

  if (listaIngles==NULL && listaEspanol==NULL)
      {
    listaIngles = nodoIng;
    listaEspanol = nodoEsp;
  }else {
    // Esta función la debe realizar usted
  }
}
// Carga las palabras desde el archivo a las
    listas
void cargarDictToListas2P (char *
    nombreArchivoBin) {
  FILE *manejadorBin;
  manejadorBin = fopen (nombreArchivoBin, "r");
  TipoDict dictTmp;
  int longitudBytes = sizeof (TipoDict);

  while (fread (&dictTmp, longitudBytes, 1,
      manejadorBin) == 1) {
    addListas2P (dictTmp.palIngles, dictTmp.
        palEspanol);
  }
  fclose (manejadorBin);
}
```

5. Ejercicios

Como se implementarían las siguiente funciones:

- **void insertarLista2P (char *palIngles, char *palEspanol)**: inserta de forma ordenada las dos palabras en sus respectivas listas.
- **void imprimirOrdenadoEspanol ()**: imprime el diccionario de forma ordenada por las palabras en español.
- **void guardarSubArchivoTexto (char *nomArchivo, char *palInicial, char *palFinal):** guarda en un archivo texto *nomArchivo* las palabras del diccionario que son mayores lexicográficamente a la cadena *palInicial* y menores a la cadena *palFinal*.

6. Recursos (`https://github.com/lgarreta/labprog/examenes`)

- La comparación con **strcmp** tiene problemas con las palabras que tienen tilde, entonces he creado un nuevo diccionario binario llamado "***palabrasBin2P.bin***" *(y minidict.txt)* sin esas palabras que llevan tilde.
- También he creado un archivo pequeño para hacer pruebas con pocas palabras y se llama "***minidict.bin***" con su respectivo archivo texto "***minidic.txt***".
- También he creado un programa para comparar cadenas: "**compararCadenas.c**" que recibe las dos cadenas e imprime el valor de **strcmp.**